Information Systems 84 (2019) 88–110

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Relational database schema design for uncertain data☆

Sebastian Link[a,*], Henri Prade[b]

[a] Department of Computer Science, University of Auckland, New Zealand
[b] IRIT, CNRS and Université de Toulouse III, France

HIGHLIGHTS

- Uncertainty in data is modeled by assigning to records a degree of possibility.
- Certainty degrees of functional dependencies indicate on which records they hold.
- Different degrees of data redundancy can then be defined and observed.
- Refined normal forms avoid or minimize data redundancy up to a target degree.
- Certainty degrees control classical trade-offs to optimize relational schema design.

ARTICLE INFO

Article history:
Received 6 November 2016
Received in revised form 24 January 2019
Accepted 2 April 2019
Available online 11 April 2019
Recommended by Jan Van den Bussche

Keywords:
Boyce–Codd normal form
Database schema design
Data redundancy
Functional dependency
Normalization
Qualitative reasoning
Third normal form
Uncertain data

ABSTRACT

Driven by the dominance of the relational model, we investigate how the requirements of applications on the certainty of functional dependencies can improve the outcomes of relational database schema design. For that purpose, we assume that tuples are assigned a degree of possibility with which they occur in a relation, and that functional dependencies are assigned a dual degree of certainty which says to which tuples they apply. A design theory is developed for functional dependencies with degrees of certainty, including efficient axiomatic and algorithmic characterizations of their implication problem. Naturally, the possibility degrees of tuples bring forward different degrees of data redundancy, caused by functional dependencies with the dual degree of certainty. Variants of the classical syntactic Boyce–Codd and Third Normal Forms are established. They are justified semantically in terms of eliminating data redundancy and update anomalies of given degrees, and minimizing data redundancy of given degrees across all dependency-preserving decompositions, respectively. As a practical outcome of our results, designers can simply fix the degree of certainty they target, and then apply classical decomposition and synthesis to the set of functional dependencies whose associated degree of certainty meets the target. Hence, by fixing the certainty degree a designer controls which integrity requirements will be enforced for the application and which data will be processed by the application. The choice of the certainty degree also balances the classical trade-off between query and update efficiency on future database instances. Our experiments confirm the effectiveness of our control parameter, and provide original insight into classical normalization strategies and their implementations.

1. Introduction

According to a Gartner forecast, the NoSQL market will be worth 3.5 billion US dollars annually by 2018, but by that time the market for relational database technology will be worth more than ten times that number, namely 40 billion US dollars annually [19]. This underlines that relational databases are here to stay, and that there is no substitute for important data [19]. However, relational databases were developed for applications in which data occur with full certainty, such as accounting, inventory, and payroll [14]. Modern applications such as information 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 [51]. Here, we establish a framework that uses information about the uncertainty in data to design more targeted relational database schemata. Indeed, the framework produces schema designs based on the qualitative degree of uncertainty in data that a target application requires. For different choices of a qualitative degree,

☆ Some results of this article have been announced in Link and Prade (2016). The research is supported by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand.

* Corresponding author.
E-mail addresses: s.link@auckland.ac.nz (S. Link), prade@irit.fr (H. Prade).

https://doi.org/10.1016/j.is.2019.04.003
0306-4379/

Table 1
Integrated relation r.

Project	*Time*	*Manager*	*Room*	*Trust*
Eagle	Mon, 9 am	Ann	Aqua	High
Hippo	Mon, 1 pm	Ann	Aqua	High
Kiwi	Mon, 1 pm	Pete	Buff	High
Kiwi	Tue, 2 pm	Pete	Buff	High
Lion	Tue, 4 pm	Gill	Buff	High
Lion	Wed, 9 am	Gill	Cyan	High
Lion	Wed, 11 am	Bob	Cyan	Good
Lion	Wed, 11 am	Jack	Cyan	Moderate
Lion	Wed, 11 am	P am	Lava	Moderate
Tiger	Wed, 11 am	P am	Lava	Low

the corresponding output designs support different levels of data integrity, data losslessness, query efficiency and update efficiency.

As a running example, sufficiently simple to motive our research and explain our findings, we consider an employee who extracts information from the web about 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 keeps track of the sources from which tuples in her integrated data set originate. Tuples from the official project meeting's web site are denoted by s_1, tuples from a project manager's web site are denoted by s_2, tuples from a project member's web site are denoted by s_3, and tuples that originate from blogs are denoted by s_4. The different sources have varying degrees of trust that people would associate with them, and typically these degrees of trust are ranked. Here, the trust in s_1 is labeled as **high**, the trust in s_1 is labeled as **good**, the trust in s_3 is labeled as **moderate**, and the trust in s_4 is labeled as **low**. Naturally, if a tuple originates from multiple sources with potentially varying degrees of trust, then we choose the maximum degree of trust associated with any of those sources. Table 1 shows the integrated relation, denoted by r, and for each tuple the degree of trust associated with it.

In classical schema design, update inefficiencies are avoided by removing data value redundancy. The latter are caused by data dependencies that model business rules that the underlying application domain is governed by. Many redundant data value occurrences are caused by functional dependencies (FDs) [4,9,36,55]. For example, the relation r from Table 1 satisfies the FD $Manager, Time \rightarrow Room$, since every pair of tuples with matching values on *Manager* and *Time* has also matching values on *Room*. This FD represents a meaningful business rule of the domain, as no manager can be present in different rooms at the same time. However, {*Manager*, *Time*} is no key as there are different tuples with matching values on *Manager* and *Time*. Indeed, the last two tuples show that different projects may be discussed in the same room at the same time. The example describes the situation in which redundant data values occur. Given that the FD is a meaningful semantic constraint that is enforced on each instance over the schema, and knowing all data values of the relation except for the value of the last tuple on *Room* (or the value of the second to last tuple on *Room*, respectively), we can infer that this value must be *Lava*. In this sense, each occurrence of the value *Lava* on attribute *Room* is redundant [55]. Indeed, updates to a single occurrence of *Lava* to a different value would result in a violation of the FD $Manager, Time \rightarrow Room$. Classical decomposition strategies into Boyce–Codd and Third normal forms eliminate redundant data value occurrences, caused by any FDs, either completely or as far as possible among all dependency-preserving decompositions, respectively [4,7,9,10,15,26,36,55]. In this example, both strategies would lead to the two schemata

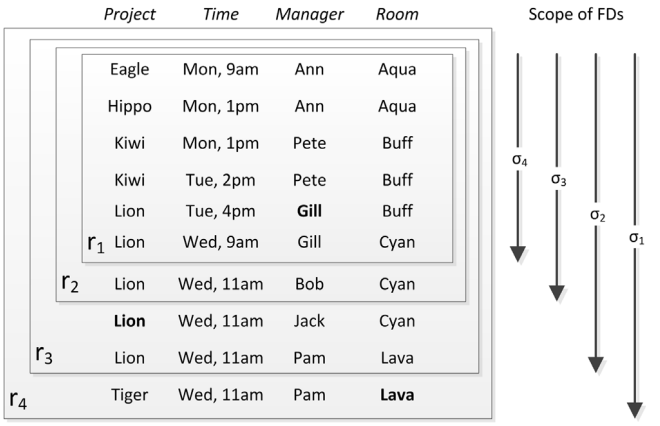

Fig. 1. Sub-relations of the relation from Table 1 together with some FDs that apply to them, and some redundant data value occurrences in bold font.

{*Time*, *Manager*, *Room*} and {*Project*, *Time*, *Manager*}, where the given FD can be enforced locally on the first schema by the key {*Time*, *Manager*}. This decomposition is illustrated on relation r from Table 1 in the form of the decomposition $\mathcal{D}_4$ in Fig. 2.

Classical relational database schema design does not take into account any information about the uncertainty of its data. In our running example, however, such information is present in the form of the degrees of trust associated with the sources that tuples originate from. The framework we develop in this article shows how such information can be used to design more targeted relational database schemata. We start from the important observation that different applications have different requirements on the quality of the data they process. In the context of uncertain data, such requirements may stipulate that data is only useful for an application if its degree of uncertainty meets a given threshold. For example, based on the degrees of trust associated with the different sources of data, different relations emerge for different applications. For every application that requires its tuples to be highly trusted, the sub-relation r_1 in Fig. 1 is the right choice. For every application that requires its tuples to have at least a good degree of trust, the sub-relation r_2 in Fig. 1 is a better choice. Similarly, other applications may favor the sub-relations r_3 or r_4, respectively, from Fig. 1. Our next important observation is that the different sub-relations are governed by different business rules, as we will exemplify on some of the FDs that holds on the sub-relations $r_1 \subseteq r_2 \subseteq r_3 \subseteq r_4$. In our example, r_4 satisfies the FD $\sigma_1 : Manager, Time \rightarrow Room$, r_3 satisfies the FD $\sigma_2 : Room, Time \rightarrow Project$, r_2 satisfies the FD $\sigma_3 : Project, Time \rightarrow Manager$, and r_1 satisfies the FD $\sigma_4 : Project \rightarrow Manager$, which we all regard as semantically meaningful constraints. Now it is important to note that FDs are *closed downwards*. That is, every FD that is satisfied by a relation, is also satisfied by every sub-relation thereof. In our example of the four FDs, r_4 satisfies $\Sigma_1 = \{\sigma_1\}$, r_3 satisfies $\Sigma_2 = \{\sigma_1, \sigma_2\}$, r_2 satisfies $\Sigma_3 = \{\sigma_1, \sigma_2, \sigma_3\}$, and r_1 satisfies $\Sigma_4 = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. This situation is illustrated in Fig. 1. The downward closure property of FDs in general, and our example in particular, stress the importance of the duality between the increase of tuples in relations and the decrease of FDs that are satisfied by them. This duality is the key idea for developing a relational schema design framework that exploits uncertainty in data. Next we use our running example as an "hors-d'oeuvre" for our framework.

Fig. 1 shows some data value occurrences in bold font. Each of these data value occurrences is redundant in the precise sense from before. Our ranking of trust degrees naturally resulted in the linear order of sub-relations, $r_1 \subseteq r_2 \subseteq r_3 \subseteq r_4$. In turn, this resulted in a linear order of FD sets, $\Sigma_1 \subseteq \Sigma_2 \subseteq \Sigma_3 \subseteq \Sigma_4$. Our main

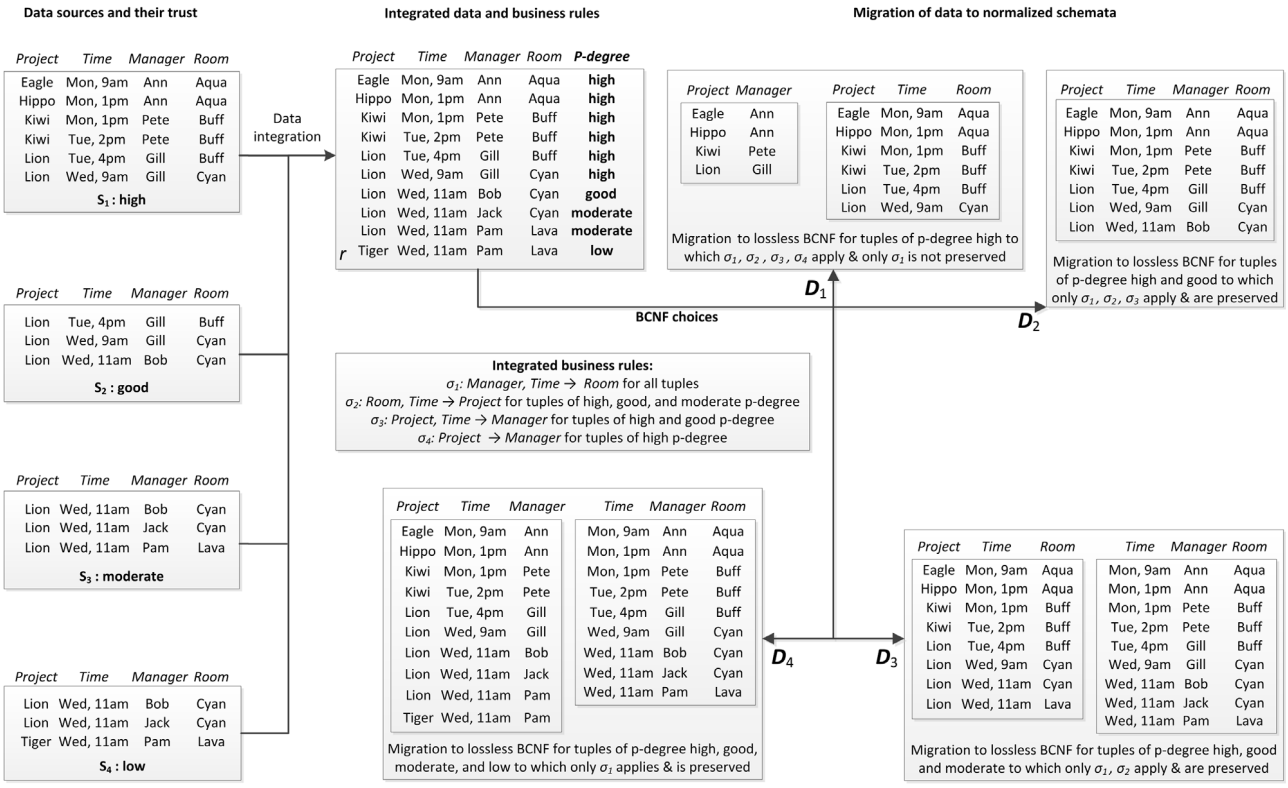

Fig. 2. Using degrees of uncertainty for data integration and tailoring schema normalization to different application requirements.

observation is that the different degrees of uncertainty allow us to define different degrees of data value redundancy. We will develop different syntactic normal forms that eliminate/minimize the corresponding degrees of data value redundancy, respectively. Indeed, redundancy in r_4 is caused by FDs in Σ_1, redundancy in r_3 is caused by FDs in Σ_2, redundancy in r_2 is caused by FDs in Σ_3, and redundancy in r_1 is caused by FDs in Σ_4. Hence, data value redundancy in r_i can be eliminated/minimized by applying classical BCNF/3NF decomposition with respect to the FD set Σ_{5-i}, for each $i = 1, \ldots, 4$, respectively. Consequently, different requirements of applications regarding their confidence in data result in different database schema designs that better accommodate these requirements.

In particular, the downward closure property means that the fewer tuples we consider in a relation the more FDs will satisfy that relation, which illustrates a trade-off between data losslessness (fewer tuples) and data integrity (more FDs that apply). This has two important consequences for the schema design of applications. Indeed, the stronger the requirements of an application are for the quality of their data, the fewer tuples meet those requirements and the more functional dependencies need to be considered to remove all redundant data value occurrences. In other words, the removal of redundant data value occurrences from relations with additional tuples requires normalization with respect to a smaller number of FDs. In turn, less normalization results in better query efficiency, as less joins are required. Note the difference to the classical normalization framework in which all applications require the same data, namely all certain tuples. This classical framework occurs as the special case of our framework in which only one degree of (un)certainty is considered. The next section provides an overview of the content of the paper, indicating also how its technical part will organized.

2. Overview

In summary, we will develop a qualitative model for uncertain data that exploits our previous observations for the benefit of relational database schema design. The model brings forward a greater variety of normalized schema designs from which the best match for the requirements of an application in terms of data integrity, data losslessness, update efficiency and query efficiency can be chosen. This will be further explained at the end of this section, and is also illustrated in Fig. 3. Our contributions are based on the following notion of a *possibilistic functional dependency* that the authors of this article introduced in [39]. Before describing our contributions we first explain this notion and the qualitative model of uncertainty.

We model uncertainty in data by assigning to each tuple $t \in r$ of a given finite relation r, a degree of possibility (p-degree), $\alpha_i = Poss_r(t)$, by which t is perceived to occur in r. A p-relation is a pair $(r, Poss_r)$ with a finite relation r and a function $Poss_r$. We refer to "degree of possibility" since the degrees are grading the possibility that the associated tuple belongs to a possible world that is compatible with the uncertain database. A more detailed analysis can be found in [39]. Our p-degrees form a finite, strictly linear chain $\alpha_1 > \cdots > \alpha_k > \alpha_{k+1}$, where α_1 is the top p-degree and α_{k+1} is the bottom p-degree, reserved for tuples that are regarded as impossible to occur in a given relation. This provides us with a discrete linear model of uncertainty, similar to the continuous probabilistic model [0,1] in which 1 denotes the top probability and 0 the bottom probability reserved for tuples that are impossible to occur. Our discrete model meets well the intuition and ability of people to reason qualitatively, without the need to quantify an exact value such as a probability. Our running example conforms to this model as the tuples from source s_i receive the p-degree α_i, for $i = 1, \ldots, 4$, and any other tuple that could be formed from values of the attributes'

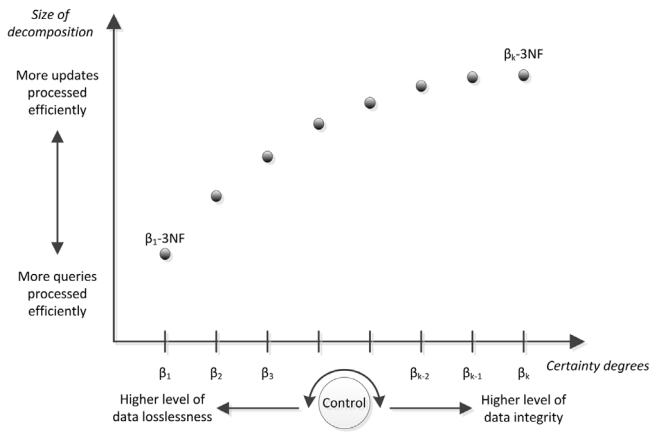

Fig. 3. Exploring certainty degrees as a mechanism to control levels of data integrity, data losslessness, query efficiency, and update efficiency.

domains and does not occur in the integrated relation r has p-degree α_5. This generalizes the *closed world assumption*, saying that tuples known not to attain any non-bottom p-degree are also known to attain the bottom p-degree. Here, the p-degrees may originate from linguistic interpretations, as in our example, **high** > **good** > **moderate** > **low** > **none**. Our model results in a nested chain of possible worlds, $r_1 \subseteq \cdots \subseteq r_k$, each of which is a classical relation that contains tuples with p-degree α_i or higher (that is, smaller i). The smallest world, r_1, contains only tuples that are fully possible to occur, while the largest world, r_k, contains all tuples but those with bottom p-degree α_{k+1}. Only tuples that occur in the smallest world, r_1, are regarded as certain, since they occur in all of the possible worlds $r_1, \ldots, r_k$. Hence, the case where $k = 1$ corresponds to the classical relational model of data, and our data model is therefore a proper extension of this model. In [39], *possibilistic functional dependencies* (pFDs) were introduced by assigning a degree of certainty (c-degree) to a classical FD. Our c-degrees form a finite, strictly linear chain $\beta_1 > \cdots > \beta_k > \beta_{k+1}$. An FD holds with c-degree β_i whenever it is satisfied in the possible world r_{k+1-i}. On the extreme ends we have FDs that hold with the top c-degree β_1 as they are satisfied in the largest possible world, r_k, and we have FDs that only hold with c-degree β_{k+1} as they are not even satisfied by the smallest possible world r_1. In our running example, σ_1 holds with c-degree β_1, σ_2 holds with c-degree β_2, σ_3 holds with c-degree β_3, and σ_4 holds with c-degree β_4, as illustrated in Fig. 1. Thus, the c-degree of a pFD becomes smaller when the violation of the pFD originates from a tuple with higher p-degree. This duality between p-degrees and c-degrees is the same as in possibilistic logic [22] where a logical proposition is all the less certain as there exists a counter-model with a high possibility (in agreement with the duality possibility/necessity in possibility theory).

The work in [39] also describes the novelty of the notion of a pFD over previous notions. In brief, other work on possibility theory associates attribute values with possibility distributions or c-degree to data, while the model in [39] assigns p-degrees to data and c-degrees to constraints. In the current article we will show that this is a right notion to exploit uncertainty in data for relational database schema design. Our contributions are as follows:

(1) Reasoning about pFDs. We establish a full design theory for pFDs, including axiomatic and algorithmic characterizations of their implication problem. Our possibilistic model allows us to develop strong links between pFDs and classical FDs. We show that a pFD $(X \rightarrow Y, \beta_i)$ with c-degree β_i is implied by a given pFD set Σ if and only if the FD $X \rightarrow Y$ is implied by the β_i-cut, Σ_{β_i}, which is the FD set containing all FDs $U \rightarrow V$ such that some pFD $(U \rightarrow V, \beta_j)$ is in Σ where β_j is of c-degree β_i or higher (that is, $j \leq i$).

In our running example, the set $\Sigma = \{(Manager, Time \rightarrow Room, \beta_1), (Time, Room \rightarrow Project, \beta_2)\}$ implies the pFD $(Manager, Time \rightarrow Project, \beta_2)$ since the FD set $\Sigma_{\beta_2} = \Sigma_2 = \{\sigma_1, \sigma_2\}$ implies the FD $Manager, Time \rightarrow Project$, but Σ does not imply the pFD $(Manager, Time \rightarrow Project, \beta_1)$ since the FD set $\Sigma_{\beta_1} = \Sigma_1 = \{\sigma_1\}$ does not imply the FD $Manager, Time \rightarrow Project$.

This result is used to establish the soundness and completeness of an axiom system for pFDs that is equivalent to Armstrong's axioms [5] for classical FDs in the special case where $k = 1$. The result also enables us to decide the implication problem for the class of pFDs in time linear in the input. Just as the classical results on Armstrong's axioms and the linear-time decidability of FDs [5,6,18] form a foundation for relational normalization using Boyce–Codd and Third normal forms, our findings form a foundation for the normalization framework we develop in the remainder of the article. As a side remark, it is also interesting to note that pFDs can be understood as a fragment of conditional functional dependencies, whose purpose is to manage the quality of certain data but not to design schemata for uncertain data [30]. The implication problem of general conditional functional dependencies is also coNP-complete to decide [30], and their consistency problem is NP-complete [30] while every set of FDs and every set of pFDs is consistent. In the same way FDs constitute a fragment of conditional functional dependencies that is used for relational database design for certain data, pFDs constitute a fragment of conditional functional dependencies that is useful for relational database design for uncertain data.

(2) Qualitative data value redundancy. The intrinsic link between the c-degree of pFDs and the p-degree of tuples in our p-relations, enables us to define different degrees of data value redundancy. In fact, a data value occurrence in some given p-relation that satisfies a given pFD set is α_i-redundant whenever *any* modification of this value to a different value results in a p-relation that violates some given pFD whose c-degree is β_{k+1-i} or higher. On the extreme ends, α_k-redundancy can only be caused by pFDs with c-degree β_1, while α_1-redundancy can be caused by any given pFD. In fact, our notion of α_i-redundancy captures precisely the impact of uncertainty on relational database schema design we intended: the smaller the p-degree of tuples with α_i-redundant data value occurrences, the smaller the number of pFDs that can cause it, and therefore the smaller the normalization effort to eliminate it. Note that actual updates only take place on relations over the relation schemata that have been established for the given application. Specifically, data in these relations is understood to meet the application requirements in terms of their p-degrees. If an update results in a tuple that does not meet the required p-degree, then the update represent a tuple deletion. On the level of the applications we have relational databases. In particular, there is no more distinction between the different p-degrees of tuples.

For our running example, Fig. 1 shows some examples of data value occurrences that are redundant in this sense, marked in bold font. For instance, *Lava* is α_4-redundant because any different value in its place would violate the FD σ_1 that holds with c-degree β_1; *Lion* is α_3-redundant because any different value in its place would violate the FD σ_2 that holds with c-degree β_2; and *Gill* is α_1-redundant because any different value in its place would violate the FD σ_4 that holds with c-degree β_4.

Given the notion of α_i-redundant data values, we define a relation schema with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$ to be in α_i-*Redundancy Free Normal Form* (RFNF) with respect to a given set of pFDs if and only if there is no p-relation over the schema

that satisfies all given pFDs and in which a data value occurrence is α_i-redundant. Again, the special case where $k = 1$ captures the definition of a RFNF in the relational model of data [55].

(3) Qualitative update anomalies. Similar to defining different degrees of data value redundancy, our framework also allows us to define different degrees of update anomalies. In fact, an α_i-update anomaly occurs whenever a p-relation that satisfies a given pFD set can be updated in such a way that all minimal keys with c-degree β_{k+1-i} or higher are satisfied after the update but not all given pFDs with c-degree β_{k+1-i} or higher. In the case of pFDs, update anomalies may occur in the form of insertions, or different kinds of modifications, but not in the form of deletions. For modifications we distinguish three different types: those in which the modified tuple does not need to satisfy any requirement (type 1), those in which the modified tuple must not be modified on the attributes of some minimal key (type 2), and those in which the modified tuple must not be modified on the attributes of the primary key (type 3).

As an example, consider the tuple with the boldly marked occurrence of **Gill** in Fig. 1. Modifying this value to **Rob** is an example of an α_1-key-based type-3 modification anomaly. The modification satisfies all minimal keys {*Manager*, *Time*}, {*Room*, *Time*}, {*Project*, *Time*} with respect to the FD set {*Manager*, *Time* $\rightarrow$ *Room*; *Room*, *Time* $\rightarrow$ *Project*; and *Project* $\rightarrow$ *Manager*}, but violates the FD *Project* $\rightarrow$ *Manager*. If {*Project*, *Time*} denotes the primary key, then the original and modified tuples agree on the primary key, which gives us an α_1-key-based type-3 modification anomaly.

We then define a relation schema with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$ to be in *α_i-Key-based Insertion Anomaly Normal Form/Key-based Type-1/2/3 Modification Anomaly Normal Form* (KIANF/KMANF-1/2/3) with respect to a given set of pFDs if and only if there is no p-relation over the schema that satisfies all given pFDs and in which a corresponding α_i-update anomaly can occur. Again, the special case where $k = 1$ captures the definitions of the normal forms in the relational model of data [55].

(4) Qualitative Boyce–Codd normal form. Our next contribution is to derive a syntactic characterization of schemata that are in α_i-RFNF with respect to the given set of pFDs, just like Boyce–Codd normal form is the syntactic normal form that captures the semantic RFNF [55]. For this purpose, we define that a given relation schema R with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$ is in β_i-BCNF with respect to a given set of pFDs, Σ, if and only if for every non-trivial pFD $(X \rightarrow Y, \beta_i)$ that can be inferred from Σ by our axioms, it is the case that $(X \rightarrow R, \beta_i)$ can be inferred from Σ by our axioms. Note that this definition is completely syntactic, utilizing our sound and complete axiomatization of pFDs. The definition also ensures that the normal form is cover-insensitive, that is, it does not depend on the choice of a representation system for pFDs. However, this leaves the question whether one can check efficiently that a given schema is in β_i-BCNF with respect to a given set Σ of pFDs. Indeed, we show that it suffices to check for all non-trivial pFDs $(X \rightarrow Y, \beta_j)$ in Σ where $j \leq i$ that $(X \rightarrow R, \beta_i)$ is implied by Σ. This condition can be validated in time quadratic in the size of Σ by our result about the linear-time decidability of pFDs. We then show the important result that a given schema is in α_i-RFNF with respect to a pFD set Σ if and only if it is in β_{k+1-i}-BCNF with respect to Σ. We also show the important result that a given schema is in α_i-KIANF/KMANF-1/2/3 with respect to a pFD set Σ if and only if it is in β_{k+1-i}-BCNF with respect to Σ. Therefore, we can validate, at design time, in quadratic time in the input whether a given schema only admits p-relations, at run time, that are free from α_i-redundant data value occurrences and free from α_i-key based update anomalies. Note that these results subsume the classical finding that a schema in BCNF only permits relations that are free from redundant data value occurrences and free from update anomalies as the special case in which $k = 1$ [55].

(5) Qualitative third normal form. In the classical normalization framework there is a trade-off between dependency-preservation, achieved by synthesizing schemata into Third normal form (3NF), and the elimination of data value redundancy, achieved by BCNF decompositions [4,9,36]. Here, dependency -preservation guarantees that all FDs can be enforced locally, without the need of joining relations to check for consistency of update operations. While a decomposition in BCNF cannot always preserve all given FDs but ensures that no data value redundancy occurs, a 3NF synthesis does preserve all given FDs but cannot always guarantee that all data value redundancy is eliminated. In fact, using well-defined measures from information theory, it has been shown [4,36] that 3NF admits a minimal amount of data value redundancy amongst all decompositions that are dependency-preserving. For these reasons, we also extend the classical results on the 3NF to our framework. We define that a given relation schema R with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$ is in *β_i-3NF* with respect to a given set of pFDs, Σ, if and only if for every non-trivial pFD $(X \rightarrow A, \beta_i)$ that can be inferred from Σ by our axioms, it is the case that $(X \rightarrow R, \beta_i)$ can be inferred from Σ by our axioms or the attribute A is β_i-prime with respect to Σ. The attribute A is β_i-prime with respect to Σ if and only if A belongs to some attribute set $X \subseteq R$ that is minimal under set containment with the property that $(X \rightarrow R, \beta_i)$ can be inferred from Σ by our axioms. Again, this definition is completely syntactic, utilizing our sound and complete axiomatization of pFDs, and also ensures that the normal form is cover-insensitive. We also show that it suffices to check for all non-trivial $(X \rightarrow Y, \beta_j)$ in Σ where $j \leq i$ that $(X \rightarrow R, \beta_i)$ is implied by Σ or every $A \in Y - X$ is β_i-prime with respect to Σ. Just as in the classical case, the latter condition is likely to be intractable to decide, due to the NP-completeness of the *prime attribute problem*, which is to decide whether a given attribute is (β_i-)prime with respect to a given set of (p)FDs [42]. Again, note that our results subsume the classical findings as the special case where $k = 1$ [10].

(6) Qualitative normalization. Next we address the goal of making our framework applicable to the design of relational database schemata. This involves two critical steps. Firstly, we show the three results that a given relation schema R with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$ is in α_i-RFNF (β_i-BCNF, β_i-3NF, respectively) with respect to the pFD set Σ if and only if the relation schema R is in RFNF (in BCNF, 3NF, respectively) with respect to the β_i-cut Σ_{β_i}. Secondly, we show that a p-relation over relation schema R and p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$ that satisfies a pFD $(X \rightarrow Y, \beta_i)$ is *α_{k+1-i}-lossless*, that is, the possible world r_{k+1-i} of that p-relation is the lossless join of its projections on XY and $X(R - Y)$: $r_{k+1-i} = r_{k+1-i}[XY] \bowtie r_{k+1-i}[X(R - Y)]$. These results together mean the following. Given a relation schema R with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$, a pFD set Σ, and a c-degree β_i with $1 \leq i \leq k$, a classical BCNF-decomposition of R with respect to the β_i-cut Σ_{β_i} gives us a decomposition that is α_{k+1-i}-lossless and free from α_{k+1-i}-redundancy. Moreover, given a relation schema R with p-degree scale $\alpha_1 > \cdots > \alpha_{k+1}$, a pFD set Σ, and a c-degree β_i with $1 \leq i \leq k$, a classical 3NF-synthesis of R with respect to the β_i-cut Σ_{β_i} gives us a decomposition that is α_{k+1-i}-lossless, β_i-dependency-preserving and only admits minimal amounts of α_{k+1-i}-redundancy amongst all β_i-dependency-preserving decompositions. We conclude that the choice of the c-degree β_i as input provides us with a mechanism to control the trade-offs between data integrity and data losslessness, and between update efficiency and query efficiency. Indeed, higher levels of data integrity are achieved by selecting lower c-degrees β_i (that is, higher i), resulting in decompositions

that target the local enforcement of the data integrity for all FDs with c-degree β_i or higher, but can only guarantee losslessness for tuples with p-degree α_{k+1-i} or higher. Vice versa, higher levels of data losslessness are achieved by selecting higher c-degrees β_i (that is, lower i), resulting in decompositions that preserve all tuples with p-degree α_{k+1-i} or higher, but only target the local enforcement of the data integrity for FDs with c-degree β_i or higher. In fact, the choice of β_i determines which FDs are used to perform classical normalization and thereby determines the number of relation schemata in the output decomposition. Heuristically, the smaller the number of relation schemata in the output, the fewer joins are required to process queries and the more efficient query processing becomes, but the less efficient update processing becomes. In reality, of course, the choice of the output design is based on which queries and updates are considered to be important. Nevertheless, the availability of different c-degrees β_i enables us to produce a greater variety of schema designs one can choose from. For a fixed choice of a c-degree β_i, the classical properties of BCNF decompositions, 3NF synthesis and their trade-offs are experienced on the level of the β_i-cut.

Fig. 2 shows BCNF design choices for different application requirements in our running example. Decomposition $\mathcal{D}_1$ is targeted at applications that only permit tuples of the highest p-degree. Consequently, all four of the FDs must be used for normalization. $\mathcal{D}_1$ avoids all α_1-redundancy but is only α_1-lossless. The decomposition did not preserve the FD σ_4. This is simply to illustrate the case that typically occurs in practice where we could not find a BCNF-decomposition that preserves all FDs. Decomposition $\mathcal{D}_2$ is targeted at applications that permit tuples of p-degree α_1 and α_2. Here, the given schema is already in β_2-BCNF, so no decomposition is required. Indeed, $\mathcal{D}_2$ avoids all α_2-redundancy and is α_2-lossless, since the FDs σ_1, σ_2 and σ_3 were used for normalization. Obviously, all of these three FDs are preserved. Decomposition $\mathcal{D}_3$ is targeted at applications that require their tuples to have p-degree α_1, α_2, or α_3. $\mathcal{D}_3$ avoids all α_3-redundancy and is α_3-lossless, since the FDs σ_1 and σ_2 were used for normalization. The decomposition preserved both of these FDs. Finally, decomposition $\mathcal{D}_4$ targets applications that accepts tuples with any non-bottom p-degree. $\mathcal{D}_4$ avoids all α_4-redundancy and is α_4-lossless, since only the FD σ_1 was used for normalization. Of course, the decomposition preserved this FD.

Another useful view of our normalization framework is the following. The choice of the target c-degree β_i means the resulting output design is classically normalized with respect to all FDs whose c-degree is β_i or higher, but may be classically de-normalized with respect to any FD whose c-degree is lower than β_i. This view suggests that our framework unifies classical normalization and de-normalization. Thereby, classically de-normalized schemata can also be justified in terms of the levels of data integrity and losslessness that are targeted.

(7) Implementation and experiments. We have implemented our algorithms and made the implementation publicly accessible by a GUI. While we view this part of our work not as intellectually challenging, we think that it is the most important contribution in terms of knowledge transfer. Extensive experiments on a distributed high-performance computing cluster confirm the effectiveness of the c-degree β_i as a control parameter for relational database schema normalization. For example, by decreasing β_i towards the bottom c-degree β_k, the normalization effort becomes gradually larger in terms of both time and the number of relation schemata in the output. This is rather natural, as the number of FDs increases. However, we also observe a saturation point for which the number of relation schemata in the output decreases again. This saturation point occurs when the presence of sufficiently many additional FDs transforms attribute sets into keys. Importantly, the different decompositions that result from different choices of β_i provide different choices in terms of update and query efficiency. The output decompositions also have precise characteristics in terms of data losslessness, data redundancy, and dependency-preservation. An organization can therefore choose the best fit for a target application by taking into account these different choices of decompositions and their characteristics. This is illustrated by the different BCNF design choices for our running example in Fig. 2. Our experiments also provide new insight into classical normalization trade-offs. For example, we provide first empirical evidence that, on average, 3NF synthesis has a fair chance of producing an optimal decomposition, that is, a lossless, dependency-preserving BCNF decomposition, while BCNF decompositions do not have a reasonable chance to be dependency-preserving. This empirical finding supports the preference for 3NF synthesis in database practice over BCNF decomposition. Designers prefer 3NF synthesis because it ensures that logically related attributes remain together while BCNF decomposition does not ensure that. Our finding suggests that 3NF synthesis also provides a significantly better chance of deriving an optimal decomposition. Our experimental results further illustrate different strengths and weaknesses of various classical versions of BCNF decomposition and 3NF synthesis algorithms. For example, the classical BCNF algorithm that requires exponential time and space [3] produces decompositions that require significantly fewer relation schemata than the BCNF algorithm that runs in polynomial time [53]. Heuristically, this suggests to apply the exponential time and space algorithm whenever feasible, since its output design will support the efficient processing of more queries than the output design of the polynomial time algorithm, simply because the latter output requires more joins. Another observation is that the 3NF algorithm that is based on a canonical cover produces fewer schemata than the 3NF algorithm that is based on a non-redundant, L-reduced cover in which all FDs have a singleton attribute on their right-hand side. This is intuitive as a canonical cover contains fewer FDs. The difference in running time of both algorithms is only marginal. Finally, we illustrate our framework on a real-world example in which we extracted information about 100 top-selling books from five online retailers, and assigned a p-degree α_i to tuples if the tuple occurred in $6 - i$ of the data sources. We mined the set of pFDs that hold on the resulting p-relation, and performed a variety of decompositions for the schema.

In summary, our framework for modeling uncertainty provides a mechanism to control the trade-off between data integrity and data losslessness, and the classical trade-off between update and query efficiency. Thereby, different schema designs can be computed that support applications with different thresholds on the confidence in the data. This is illustrated in Fig. 3.

Organization. Our data model of uncertainty is presented in Section 3, and the notion of a pFD is given in Section 4. The design theory of pFDs is established in Section 5. Qualitative variants of BCNF and 3NF are defined in Section 6, and their semantic justifications are derived. Normalization algorithms are established in Section 7. The GUI is briefly outlined in Section 8. Experimental results are discussed in Section 9. Related work is discussed in Section 11. A real-world example from Web data extraction is analyzed in Section 10. Finally, Section 12 concludes and comments on future work.

3. Uncertain databases

We recall the possibilistic data model introduced in [39]. For a discussion related to the possibilistic grounding of our model and its originality in comparison to previous work we refer the interested reader to [39]. The main motivation for the possibilistic data model its use of the downward closure property of functional dependencies to apply relational database schema design

for applications with uncertain data. This is the contribution of the current article.

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 a running example we use the relation schema MEETING with attributes *Project*, *Time*, *Manager*, and *Room*, that we already introduced in the introductory section.

In the classical relational data model, tuples either belong or do not belong to a relation, so 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 11 am in room Cyan than we have confidence that Gill attends a meeting of project Lion on Wednesday at 11 am in room Cyan. Effective database support for new applications, such as data cleaning and integration, requires us to accommodate uncertainty in data and to make the most out of this additional information. In general, one may distinguish between uncertainty at the attribute level, and uncertainty at the tuple level. In practice, the choice would depend on which information is available.

We define possibilistic 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*, that is, a finite, strictly 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* (p-degrees). The top p-degree α_1 is reserved for tuples that are 'fully possible' to occur in a relation, while the bottom p-degree α_{k+1} is reserved for tuples that are 'not possible at all', that is 'impossible', to occur in a relation. The use of the bottom p-degree α_{k+1} in our possibilistic data model is the counterpart of the classical closed world assumption. 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, that is, where $k = 1$.

In our running example, the employee classifies the possibility with which tuples in her integrated data set occur according to the source of the information. Tuples from the official project meeting's web site are assigned p-degree α_1, 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 blogs are assigned degree α_4. Implicitly, any other tuple has degree α_5, indicating that it is impossible to occur according to the current state of information. For a recent sophisticated method to assign degrees of trusts with websites see Google's approach from [20]. If desired, the p-degree may carry some linguistic interpretation such 'fully possible' > 'quite possible' > 'medium possible' > 'somewhat possible' > 'not possible at all'. The scale could have also originated from a different interpretation, say already held meetings are classified as α_1, confirmed meetings as α_2, requested meetings as α_3, planned meetings as α_4, and all other meetings as α_5. The degrees may also originate from numerical interpretations, such as $1 > 0.75 > 0.5 > 0.25 > 0$. Either way, the employee has chosen five p-degrees $\alpha_1 > \alpha_2 > \alpha_3 > \alpha_4 > \alpha_5$ to qualitatively assign different degrees of uncertainty to tuples, with top p-degree α_1 and bottom p-degree α_5. If information is available on the attribute level, we could still convert this information to the tuple level. In this case, it would make sense to assign to a tuple the minimum p-degree across its attribute values.

Formally, a *possibilistic relation schema* (p-schema) $(R, \mathcal{S})$ consists of a relation schema R and a possibility scale $\mathcal{S}$. A *possibilistic relation* (p-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 p-degree $Poss_r(t)$ in the possibility scale $\mathcal{S}$. Sometimes, we simply refer to a p-relation $(r, Poss_r)$ by r, assuming that $Poss_r$ has been fixed. For example, Table 1 shows our p-relation $(r, Poss_r)$ over (MEETING, $\mathcal{S} = \{\alpha_1, \ldots, \alpha_5\}$).

P-relations enjoy a well-founded semantics in terms of possible worlds. In fact, a p-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 p-degree of at least α_i, that is, $r_i = \{t \in r \mid Poss_r(t) \geq \alpha_i\}$. The linear order of the p-degrees results in a 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, that is, 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 p-degree α_1 takes on a distinguished role: every tuple that is 'fully possible' occurs in every possible world – and is thus – 'fully certain'. This formally confirms our intuition that possibilistic relations subsume relations (of fully certain tuples) as a special case. Fig. 1 shows the possible worlds $r_1 \subsetneq r_2 \subsetneq r_3 \subsetneq r_4$ of our example p-relation from Table 1. For $i = 1, \ldots, 4$, the possible world r_i contains all those tuples t from r where $Poss_r(t) \geq \alpha_i$.

4. Possibilistic FDs

We recall the notion of a possibilistic functional dependency (pFDs) introduced in [39]. For a discussion on the novelty of this notion in relationship to previous work, and the equivalence of its associated implication problem to that of Horn clauses in possibilistic propositional logic, we refer the interested reader to [39]. The remainder of the current article will show that the notion of pFDs from [39] is fundamental to schema design for possibilistic data, in the same way the classical notion of FDs is fundamental to schema design for certain data.

Recall that an FD $X \to Y$ is satisfied by a relation r whenever 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 r_1, but not by r_2 and therefore not by r_3 and r_4. The FD *Project*, *Time* $\to$ *Manager* is satisfied by r_1 and r_2, but not by r_3 and therefore not by r_4. The FD *Time*, *Room* $\to$ *Project* is satisfied by 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 p-degrees of tuples that define a p-relation also define degrees of certainty with which FDs hold in the p-relation. Intuitively, since the FD *Manager*, *Time* $\to$ *Room* is satisfied in every possible world, it is fully certain to hold in r. As the FD *Time*, *Room* $\to$ *Project* is only violated in a somewhat possible world r_4, it is quite certain. Since the FD *Project*, *Time* $\to$ *Manager* is only violated in a medium possible world r_3, it is medium certain. As the FD *Project* $\to$ *Manager* is only violated in a quite possible world r_2, it is somewhat certain. Finally, as the FD *Manager*, *Room* $\to$ *Time* is violated in the fully possible world r_1, it is not certain at all.

In summary, the marginal certainty with which an FD holds in a p-relation corresponds to the possibility degree of the smallest possible world in which the FD is violated. This is illustrated in Fig. 4 on our running example. Therefore, similar to a scale $\mathcal{S}$ of possibility degrees for tuples we use a scale $\mathcal{S}^T$ of certainty degrees (c-degrees) for FDs. We commonly use subscripted versions of the Greek letter β to denote c-degrees associated with

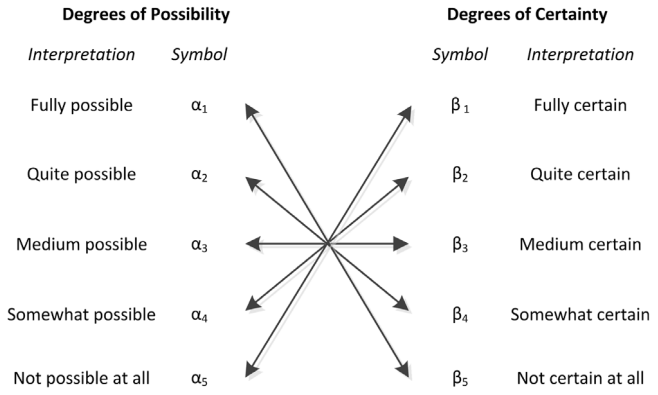

Fig. 4. The p-degree α_i of the smallest world r_i that violates an FD σ determines the marginal c-degree β_{k+2-i} by which σ holds on the p-relation r.

FDs. Formally, the duality between p-degrees in $\mathcal{S}$ and the c-degrees in $\mathcal{S}^T$ can be defined by the mapping $\alpha_i \mapsto \beta_{k+2-i}$, for $i = 1, \ldots, k+1$. Assuming that the world r_{k+1} cannot satisfy any FD, the marginal certainty $C_{(r,Poss_r)}(X \to Y)$ with which the FD $X \to Y$ holds on the p-relation $(r, Poss_r)$ is the c-degree β_{k+2-i} that corresponds to the smallest world r_i in which $X \to Y$ is violated, that is,

$$C_{(r,Poss_r)}(X \to Y) = \min\{\beta_{k+2-i} \mid \not\models_{r_i} X \to Y\}.$$

In particular, if r_k satisfies $X \to Y$, then $C_{(r,Poss_r)}(X \to Y) = \beta_1$. We can now define the syntax and semantics of pFDs.

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

The p-relation $(r, Poss_r)$ of our running example satisfies: $(Manager, Time \to Room, \beta_1)$, $(Room, Time \to Project, \beta_2)$, $(Project, Time \to Manager, \beta_3)$, $(Project \to Manager, \beta_4)$, and $(Manager, Room \to Time, \beta_5)$. The latter pFD is trivial as β_5 is here the bottom c-degree, meaning 'not certain at all'. Since $C_{(r,Poss_r)}(Project, Time \to Manager) = \beta_3 < \beta_2$, the p-relation violates the pFD $(Project, Time \to Manager, \beta_2)$.

PFDs form a class of integrity constraints tailored to possibilistic data. Indeed, a pFD $(X \to Y, \beta_i)$ separates semantically meaningful from meaningless p-relations by allowing violations of the FD $X \to Y$ only by tuples with a p-degree α_j where $j \leq k + 1 - i$. For $i = 1, \ldots, k$, the c-degree β_i of $(X \to Y, \beta_i)$ means that the 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 data integrity effectively. The other main driver for pFDs is their impact on the notion of data redundancy. As pFDs with different c-degree do not apply to the same worlds, they cause different degrees of data redundancy. As argued in the introduction already, this observation saves normalization effort when eliminating data redundancy from less possible worlds, on which only more certain FDs hold.

5. Qualitative design theory

Classical normalization is founded on the theory of functional dependencies, in particular the axiomatic and algorithmic solutions to their implication problem. Consequently, we now establish a design theory for pFDs as a foundation for normal forms, their justification, and normalization of p-schemata. First, we establish a strong link between the implication problem of pFDs and the implication problem of FDs, which is a consequence of the downward closure property of FDs. Based on this link, we then establish axiomatic and algorithmic solutions to the implication problem of pFDs. Note that our design theory for pFDs subsumes the design theory for classical FDs as the special case where $k = 1$.

5.1. β-cuts

We establish a precise correspondence between instances of the implication problem for pFDs and instances of the implication problem for traditional FDs. Let $\Sigma \cup \{\varphi\}$ denote a set of pFDs over a p-schema $(R, \mathcal{S})$. We say that Σ *implies* φ, denoted by $\Sigma \models \varphi$, if every p-relation $(r, Poss_r)$ over $(R, \mathcal{S})$ that satisfies every pFD in Σ also satisfies φ.

Example 2. Let Σ consist of the following four pFDs

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

over (MEETING, $\{\alpha_1, \ldots, \alpha_5\}$). Further, let φ denote the pFD $(Room, Time \to Manager, \beta_2)$. Then Σ does not imply φ as the following p-relation witnesses.

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

For a set Σ of pFDs on some p-schema $(R, \mathcal{S})$ and c-degree $\beta \in \mathcal{S}^T$ where $\beta > \beta_{k+1}$, let

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

be the β-*cut* of Σ. The major strength of our framework is engraved in the following result. It says that a pFD $(X \to Y, \beta)$ with c-degree β is implied by a set Σ of pFDs if and only if the FD $X \to Y$ is implied by the β-cut Σ_β of Σ.

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

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

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

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

$$C_{(r,Poss_r)}(U \to V) \geq \beta' \geq \beta. \tag{2}$$

If r_i violated $U \to V$, then

$$\begin{array}{lll} \beta & > & \beta_{k+2-i} \qquad \text{by (1)} \\ & \geq & C_{(r,Poss_r)}(U \to V) \qquad \text{by Definition of } C_{(r,Poss_r)} \\ & \geq & \beta \qquad \text{by (2)} \end{array}$$

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 $X \to Y$. Without loss of generality, 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 $(X \to Y, \beta)$

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

$\frac{}{XY \to Y}$ (reflexivity, $\mathcal{R}'$)	$\frac{X \to Y}{XZ \to Y}$ (extension, $\mathcal{E}'$)	$\frac{X \to Y \quad Y \to Z}{X \to Z}$ (transitivity, $\mathcal{T}'$)

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

$\frac{}{(XY \to Y, \beta)}$ (reflexivity, $\mathcal{R}$)	$\frac{(X \to Y, \beta)}{(XZ \to Y, \beta)}$ (extension, $\mathcal{E}$)	$\frac{(X \to Y, \beta) \; (Y \to Z, \beta)}{(X \to Z, \beta)}$ (transitivity, $\mathcal{T}$)	$\frac{}{(X \to Y, \beta_{k+1})}$ (bottom, $\mathcal{B}$)	$\frac{(X \to Y, \beta)}{(X \to Y, \beta')} \; \beta' < \beta$ (weakening, $\mathcal{W}$)

since $C_{(r,Poss_r)}(X \to Y) = \beta_{k+2-i}$, 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 $C_{(r,Poss_r)}(U \to V) = \beta_1 \geq \beta$. If r_i violates $U \to V$, then $U \to V \notin \Sigma_\beta$, i.e., $\beta' < \beta = \beta_{k+1-i}$. Therefore, $\beta' \leq \beta_{k+2-i} = C_{(r,Poss_r)}(U \to V)$ as $r_i = r'$ is the smallest relation that violates $U \to V$. We conclude that $C_{(r,Poss_r)}(U \to V) \geq \beta'$. Consequently, $(r, Poss_r)$ is a possibilistic relation that satisfies Σ and violates $(X \to Y, \beta)$. □

The following example illustrates Theorem 1.

Example 3. Let Σ and φ be as in Example 2, in particular Σ does not imply φ. Theorem 1 reduces the implication problem of pFDs to that of FDs, namely Σ_{β_2} does not imply *Room*, *Time* → *Manager*. Indeed, the possible world r_3 of the p-relation from Example 2

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

5.2. Armstrong axioms

The *semantic closure* $\Sigma^* = \{\varphi \mid \Sigma \models \varphi\}$ contains all pFDs implied by Σ. We compute Σ^* by applying *inference rules* of the form $\frac{\text{premise}}{\text{conclusion}}$ condition, where rules without premise are *axioms*. For a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote that there is an *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 in Σ or the result of applying a 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 p-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 a (finite) *axiomatization* if $\mathfrak{R}$ is both sound and complete.

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 [5]. The axioms are shown in Table 2.

PFDs enjoy the axiomatization $\mathfrak{P}$ from Table 3. It subsumes Armstrong's axioms [5] for the special case where the scale $\mathcal{S}^T$ consists of just two c-degrees. In these rules all attribute sets X, Y, Z are subsets of the given relation schema R, the c-degrees β and β' belong to the given certainty scale $\mathcal{S}^T$, and β_{k+1} denotes the bottom c-degree. 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. For a completeness proof of $\mathfrak{P}$ we could have used the classical strategy to write down a two-tuple p-relation that violates a given pFD $(X \to Y, \beta)$ that cannot be inferred from a given pFD set Σ using $\mathfrak{P}$. Instead, we establish the completeness of $\mathfrak{P}$ directly by showing that a given pFD $(X \to Y, \beta)$ that is implied by Σ can be also be inferred from Σ using the rules in $\mathfrak{P}$. This direct proof shows how the bottom axiom $\mathcal{B}$ and weakening rule $\mathcal{W}$ can be applied to reduce the inference of $(X \to Y, \beta)$ from Σ to an inference of $X \to Y$ from Σ_β. However, the completeness of $\mathfrak{A}$ guarantees immediately that $X \to Y$ can be inferred from Σ_β, due to Theorem 1 and the assumption that $(X \to Y, \beta)$ is implied by Σ.

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

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 possibilistic 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 4. Let Σ and φ be as in Example 2, and $\varphi' = (Room, Time \to Manager, \beta_3)$. We show an inference of φ' from Σ by $\mathfrak{P}$. Attributes are abbreviated by their first letters.

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

Of course, φ cannot be inferred from Σ by $\mathfrak{P}$, as Example 2 and the soundness of $\mathfrak{P}$ show.

5.3. Decision algorithm

In practice it is often unnecessary to compute the closure Σ^* from a given set Σ. Instead, rather frequently occurs the problem of deciding whether a given Σ implies a given 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

One may compute Σ^* and check if $\varphi \in \Sigma^*$, but this is inefficient and does not make effective use of the additional input φ. For pFDs, we can exploit Theorem 1 to derive a linear time algorithm that decides the implication problem. Given a pFD set $\Sigma \cup \{(X \to Y, \beta)\}$ we return *true* if $\beta = \beta_{k+1}$ (since this is the trivial case where β_{k+1} is the bottom c-degree), otherwise it is sufficient to check if $\Sigma_\beta \models X \to Y$. 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 [6]. 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. We obtain the following algorithmic characterization.

ALGORITHM 1: Closure Computation

Input: Relation schema R, attribute set $X \subseteq R$, set Σ_β of FDs over R
Output: Attribute set closure *Closure* of X with respect to Σ_β

```
Closure ← X;
FDList ← List of X → Y ∈ Σ_β;
repeat
    OldClosure ← Closure;
    Remove Closure from LHS of FDs in FDList;
    for all ∅ → Y in FDList do
        Closure ← Closure ∪ Y;
        FDList ← FDList − {∅ → Y};
    end
until Closure=OldClosure or FDList=[ ];
return(Closure);
```

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

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

We illustrate the algorithm on our running example.

Example 5. Let Σ and φ be as in Example 2, and $\varphi' = (Room, Time \to Manager, \beta_3)$. It follows that φ is not implied by Σ as $RT^+_{\Sigma_{\beta_2}} = RTP$. Furthermore, φ' is implied by Σ as $RT^+_{\Sigma_{\beta_3}} = RTPM$.

6. Qualitative normal forms

In relational schema design, Boyce–Codd normal form (BCNF) syntactically characterizes relation schemata that are guaranteed to be free of data value redundancy in all relations over the schema, in terms of FDs [55]. Third normal form (3NF) syntactically characterizes relation schemata that are guaranteed to have a minimal amount of data value redundancy in their relations amongst all schemata on which all FDs can be enforced locally [36]. In relations, data value redundancy is treated uniformly for all data, and the elimination of all data value redundancy requires normalization with respect to all FDs.

In p-relations, different tuples may have different p-degrees, and different pFDs may apply to them. Hence, data value redundancy is caused by pFDs with different c-degrees and occurs in tuples of different p-degrees. Indeed, the smaller the p-degree for which data value redundancy is to be eliminated, the smaller the number of pFDs that can cause this redundancy. Consequently, the smaller the normalization effort will be, too. We will now exploit this observation to tailor relational schema design for applications with different requirements for the uncertainty of their data. For this purpose, we will introduce notions of data value redundancy that target the p-degree of tuples in which they occur. This results in a variety of semantic normal forms by which data value redundancy of growing p-degrees are eliminated. We characterize each of the semantic normal forms by a corresponding syntactic normal form, and establish strong correspondences with BCNF and 3NF in relational databases.

6.1. Redundancy-free normal form

Motivated by the example in our introduction we propose different notions of data value redundancy that are tailored towards the different p-degrees with which tuples occur in a p-relation. For this, we exploit the classical proposal by Vincent [55]. 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 such that: i) for all $\bar{A} \in R - \{A\}$ we have $\bar{t}(\bar{A}) = t(\bar{A})$, and ii) $\bar{t}(A) \neq t(A)$. For a relation r over R that satisfies 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 [55].

Definition 6. Let $(R, \mathcal{S})$ denote a p-schema, Σ a set of pFDs over $(R, \mathcal{S})$, $A \in R$ an attribute, $(r, Poss_r)$ a p-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 value 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 c-degree $\beta_1, \ldots, \beta_k$, while α_k-redundancy can only be caused by pFDs with c-degree β_1. Naturally, we have now arrived at the following definition.

Definition 7. A p-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 p-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.

For example, (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 directly from the redundant data value occurrences in Fig. 1, but the satisfaction of the α_2-RFNF condition is not obvious. The next result shows that α_i-RFNF characterizes p-schemata that permit only p-relations whose possible world r_i is free from data redundancy caused by the classical FDs that apply to it.

Theorem 4. *$(R, \mathcal{S})$ 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 $C_{(r,Poss_r)}(U \to V) = \beta_1 \geq \beta$. If r_i violates $U \to V$, then $C_{(r,Poss_r)}(U \to V) = \beta_{k+2-i} \geq \beta$. As $t(A)$ is α_i-redundant we have shown that $(R, \mathcal{S})$ is not in α_i-RFNF with respect to Σ. □

6.2. Boyce–Codd Normal Form

Our goal is now to characterize α-RFNF, which is a semantic normal form, purely syntactically. Therefore, we propose qualitative variants of 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, $\Sigma^+_{\mathfrak{A}}$ denotes the syntactic closure of Σ with respect to the set $\mathfrak{A}$ of Armstrong's axioms [5]. While α-RFNF is defined semantically using the p-degree α of a possible world, qualitative variants of BCNF are defined syntactically using the c-degrees of the given pFDs.

Definition 8. A p-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 $(X \to R, \beta) \in \Sigma^+_{\mathfrak{P}}$.

Recall that sets Σ and Θ are *covers* of one another if $\Sigma^* = \Theta^*$ holds. 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 β-BCNF condition for a pFD set Σ can be characterized by the BCNF condition for the FD set Σ_β.

Theorem 5. *$(R, \mathcal{S})$ 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 $X \to Y \in \left(\Sigma_\beta\right)^+_{\mathfrak{A}}$ where $Y \not\subseteq X$ we have $X \to R \in \left(\Sigma_\beta\right)^+_{\mathfrak{A}}$, 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.

Theorem 6. *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 4, $(R, \mathcal{S})$ 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 5. □

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

Fig. 5. Relationships between normal forms.

Fig. 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 p-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 7. *A p-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 5, $(R, \mathcal{S})$ 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 $X \to R \in \left(\Sigma_\beta\right)^+_{\mathfrak{A}}$. 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 $X \to R \in \left(\Sigma_\beta\right)^+_{\mathfrak{A}}$ is equivalent to saying that $(X \to R, \beta') \in \Sigma^+_{\mathfrak{P}}$ for some $\beta' \geq \beta$, according to Theorems 2 and 1. □

Example 9. Let $(\textsc{Meeting}, \mathcal{S})$ and Σ be as in Example 2. Using Theorem 7 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 6 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 5 it follows that $\textsc{Meeting}$ 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 4, it follows that $\textsc{Meeting}$ 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}.

6.3. Normal forms that avoid update anomalies

We have seen that α_i-RFNF corresponds to β_{k+1-i}-BCNF, which provides a principled semantic justification of BCNF. In the relational model, there are other semantic justifications for Boyce–Codd normal form. For example, tuples usually represent entities of the real-world. These entities can be uniquely identified by the values on their key attributes. It is therefore logical to assume that any updates of an entity only require us to check whether all specified keys are still valid after the update. In other words, a good semantic normal form would avoid any update anomalies, which occur when updates result in instances that satisfy all keys but violate some other constraint. Another nice feature of such a normal form is that it integrity checking becomes less computationally involved: key validation can be done in logarithmic time using appropriate indices, while the validation of functional

dependencies requires quadratic time in the number of tuples. Our main result shows that β_{k+1-i}-BCNF characterizes those p-relation schemata that avoid α_i-insertion, and three different types of modification anomalies. Note that deletion anomalies cannot occur with functional dependencies.

6.3.1. The relational Case

We start by summarizing the formal definitions of key-based update anomalies from the relational model of data [55]. These will allow us to state the achievements of the classical Boyce–Codd normal form in terms of avoiding these anomalies.

An attribute subset X of a relation schema R is said to be a *superkey* for an FD set Σ if and only if Σ implies the FD $X \to R$. We say that X is a *candidate key* of R for Σ if and only if X is a superkey of R for Σ and there is no proper subset Y of X such that Y is a superkey of R for Σ. We use Σ^k to denote the set of candidate keys of R for Σ. A *primary key* of R for Σ is a distinguished element of Σ^k.

A relation r over relation schema R is said to have a *key-based insertion anomaly* (KIA) with respect to the set Σ of functional dependencies if and only if (i) r satisfies Σ, (ii) there is some R-tuple $t \notin r$ such that $r \cup \{t\}$ satisfies Σ^k, and (iii) $r \cup \{t\}$ violates Σ.

A relation r over relation schema R is said to have a *key-based type-1 modification anomaly* (KMA-1) with respect to the set Σ of functional dependencies if and only if there is a tuple $t \in r$ and an R-tuple $t' \notin r$ such that (i) r satisfies Σ, (ii) $(r - \{t\}) \cup \{t'\}$ satisfies Σ^k, and (iii) $(r - \{t\}) \cup \{t'\}$ violates Σ.

A relation r over relation schema R is said to have a *key-based type-2 modification anomaly* (KMA-2) with respect to the set Σ of functional dependencies if and only if there is a tuple $t \in r$ and an R-tuple $t' \notin r$ such that (i) r satisfies Σ, (ii) $(r - \{t\}) \cup \{t'\}$ satisfies Σ^k, (ii') there is some key $K \in \Sigma^k$ such that $t[K] = t'[K]$, and (iii) $(r - \{t\}) \cup \{t'\}$ violates Σ.

A relation r over relation schema R is said to have a *key-based type-3 modification anomaly* (KMA-3) with respect to the set Σ of functional dependencies if and only if there is a tuple $t \in r$ and an R-tuple $t' \notin r$ such that (i) r satisfies Σ, (ii) $(r - \{t\}) \cup \{t'\}$ satisfies Σ^k, (ii") for the primary key $K \in \Sigma^k$ we have $t[K] = t'[K]$, and (iii) $(r - \{t\}) \cup \{t'\}$ violates Σ.

(R, Σ) is said to be in *key-based insertion anomaly normal form* (KIANF) if and only if there is no relation r over R which as a KIA with respect to Σ. Likewise, for $j = 1, \ldots, 3$, (R, Σ) is said to be in *key-based modification anomaly normal form j* (KMANF$_i$) if and only if there is no relation r over R which as a KMA-j with respect to Σ.

The main result for the relational model is that Boyce–Codd normal form characterizes the relation schemata that are free from key-based insertion anomalies and key-based type-i modification anomalies for $i = 1, 2, 3$.

Theorem 8 (*[55]*). *Let Σ be a set of functional dependencies over relation schema R. Then the following are equivalent:*

- *(R, Σ) is in KIANF*
- *(R, Σ) is in KMANF-1*
- *(R, Σ) is in KMANF-2*
- *(R, Σ) is in KMANF-3*
- *(R, Σ) is in BCNF.* □

6.3.2. The possibilistic Case

We will now generalize the results of the last subsection to our possibilistic setting. Note that the relational case is subsumed by the possibilistic model as the special case where $k = 1$. We begin by distinguishing update anomalies with respect to the p-degrees of tuples in which they occur.

Definition 10. Let Σ be a set of pFDs over p-relation schema $(R, \mathcal{S})$ where $|\mathcal{S}| = k + 1$, and let $i \in \{1, \ldots, k\}$. Then a p-relation r is said to have an *α_i-insertion anomaly* (α_i-KIA) with respect to Σ if and only if r_i has an insertion anomaly with respect to Σ_{α_i}. $((R, \mathcal{S}), \Sigma)$ is said to be in *α_i-key-based insertion anomaly normal form* (α_i-KIANF) if and only if there is no p-relation r over $(R, \mathcal{S})$ such that r has an α_i-KIA with respect to Σ.

For $j = 1, 2, 3$, a p-relation r is said to have an *α_i-modification anomaly of type-j* with respect to Σ if and only if r_i has a key-based type-j modification anomaly (α_i-KMA-j) with respect to Σ_{α_i}. $((R, \mathcal{S}), \Sigma)$ is said to be in *α_i-key-based modification anomaly normal form-j* (α_i-KMANF-j) if and only if there is no p-relation r over $(R, \mathcal{S})$ such that r has an α_i-KMA-j with respect to Σ.

We illustrate the definitions by the following examples. For these examples, let Σ consist of the following four pFDs

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

over (MEETING, $\{\alpha_1, \ldots, \alpha_5\}$). In addition, r denotes the p-relation from Table 1. We start with an insertion anomaly.

Example 11. We illustrate the case of an α_4-KIA. We know that $\Sigma_{\alpha_4} = \{Manager, Time \to Room\}$, and $\Sigma^k_{\alpha_4} = \{\{Manager, Time, Project\}\}$. Now, if t denotes the tuple

(*Project*: Tiger; *Time*: Wed, 11 am; *Manager*: Pam; *Room*: Mauve), then $r_4 \cup \{(t, \alpha_4)\}$ satisfies $\Sigma^k_{\alpha_4}$, but violates Σ_{α_4}.

Next we will illustrate the case of a modification anomaly.

Example 12. We illustrate the case of an α_3-KMA-2. We know that $\Sigma_{\alpha_3} = \{Manager, Time \to Room; Room, Time \to Project\}$, and $\Sigma^k_{\alpha_3} = \{\{Manager, Time\}\}$. Now, if t denotes the tuple

(*Project*: Lion; *Time*: Wed, 11 am; *Manager*: Jack; *Room*: Cyan), and t' denotes the tuple

(*Project*: Panda; *Time*: Wed, 11 am; *Manager*: Jack; *Room*: Cyan), then $t[Time, Manager] = t'[Time, Manager]$, and $(r_3 - \{(t, \alpha_3)\}) \cup \{(t', \alpha_3)\}$ satisfies $\Sigma^k_{\alpha_3}$, but violates Σ_{α_3}.

Finally, we give another example of a modification anomaly.

Example 13. We illustrate the case of an α_1-KMA-3. We know that $\Sigma_{\alpha_1} = \{Manager, Time \to Room; Room, Time \to Project; Project \to Manager\}$, and $\Sigma^k_{\alpha_1} = \{\{Manager, Time\}, \{Room, Time\}, \{Project, Time\}\}$ with primary key $\{Project, Time\}$. Now, if t denotes the tuple

(*Project*: Lion; *Time*: Tue, 4 pm; *Manager*: Gill; *Room*: Buff), and t' denotes the tuple

(*Project*: Lion; *Time*: Tue, 4 pm; *Manager*: Rob; *Room*: Buff), then $t[Project, Time] = t'[Project, Time]$, and $(r_1 - \{(t, \alpha_1)\}) \cup \{(t', \alpha_1)\}$ satisfies $\Sigma^k_{\alpha_1}$, but violates Σ_{α_1}.

We are now able to state the main result of this subsection.

Theorem 9. *Let Σ be a set of pFDs over p-relation schema $(R, \mathcal{S})$ where $|\mathcal{S}| = k + 1$, and let $i \in \{1, \ldots, k\}$. Then the following are equivalent:*

1. *$((R, \mathcal{S}), \Sigma)$ is in α_i-KIANF*
2. *(R, Σ_{α_i}) is in KIANF*
3. *$((R, \mathcal{S}), \Sigma)$ is in α_i-KMANF-1*
4. *(R, Σ_{α_i}) is in KMANF-1*
5. *$((R, \mathcal{S}), \Sigma)$ is in α_i-KMANF-2*
6. *(R, Σ_{α_i}) is in KMANF-2*
7. *$((R, \mathcal{S}), \Sigma)$ is in α_i-KMANF-3*

8. (R, Σ_{α_i}) *is in KMANF-3*
9. $((R, \mathcal{S}), \Sigma)$ *is in* β_{k+1-i}*-BCNF.*

Proof. The result follows straight from the definitions, the fact that $\Sigma_{\alpha_i} = \Sigma_{\beta_{k+1-i}}$, and Theorem 8. □

Example 14. For our running example, we know that the p-relation schema (MEETING, $\{\alpha_1, \ldots, \alpha_5\}$) with pFD set Σ is not in α_1-BCNF, not in α_2-BCNF, and not in α_3-BCNF, but it is in α_2-BCNF. Examples 11–13 therefore also illustrate Theorem 9.

6.4. 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 FDs can be enforced locally, without the need of joining relations to check for consistency of updates.

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 p-schema $(R, \mathcal{S})$, a c-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 15. A p-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 5 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 p-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 p-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* β*-prime.*

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. □

Example 16. Let (MEETING, $\mathcal{S}$) and Σ be as in Example 2. Using Theorem 11, the schema is in neither β_1- nor β_2-3NF, but it is in β_3-3NF and in β_4-3NF with respect to Σ. Finally, by Theorem 10, 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 with respect to Σ_{β_4}.

7. Qualitative normalization

We now establish algorithmic means to design relational database schemata for applications with uncertain data. We first describe our overall strategy of database normalization for such applications, and its impact on current database design practice. Subsequently, we address decompositions into variants of Boyce–Codd normal form and synthesis into variants of Third normal form.

7.1. Strategy and impact on practical database design

The input to the database normalization process is a possibilistic schema $(R, \mathcal{S})$ together with a set Σ of possibilistic functional dependencies. As in the classical design process, data is not required for the process. In particular, we do not require any p-relations as part of the input. However, having data available is likely to benefit the design process as the data can help designers to identify those pFDs that are meaningful for the domain of our applications. For instance, there is empirical evidence that Armstrong databases do help with the acquisition of meaningful functional dependencies [37]. Recall that Armstrong databases are databases that satisfy those pFDs of the input set, and violate all those pFDs that are not implied by the input set. We enlist the investigation of Armstrong p-relations for sets of pFDs under future work, and assume that the input Σ to the database normalization process consists of those pFDs that are meaningful for the given domain.

Indeed, the availability of c-degrees as part of the input provides us with many choices for database normalization. The choice we make is determined by the application requirements. In fact, based on the requirements that applications have on the p-degrees of tuples or the c-degrees of constraints, we fix the appropriate c-degree $\beta \in \mathcal{S}^T$ that determines which possible world we classically normalize with respect to the set of classical FDs that apply to it. Note that the duality between the p-degrees and the c-degrees allows us to fix the dual c-degree, even if the application requirements are only available in terms of the p-degrees.

Once we have fixed the certainty degree β, we can simply perform classical normalization with respect to the set of classical FDs whose c-degrees are at least as high as the target c-degree, that is, with respect to the β-cut Σ_β. Indeed, we can pursue BCNF decompositions to obtain α_{k+1-i}-lossless decompositions free from any α_{k+1-i}-data value redundancy but potentially not β_i-dependency-preserving (that is, some FDs may require validation on the join of some relations). We can also pursue 3NF synthesis to obtain α_{k+1-i}-lossless, β_i-dependency-preserving decompositions where the degree of α_{k+1-i}-data redundancy is minimal with respect to all decompositions that are β_i-dependency-preserving.

Different choices of c-degrees address different application requirements. Hence, the availability of the c-degrees provides organizations with a variety of normalized database schemata. In this sense, the degree of certainty is a parameter that allows

stakeholders to control trade-offs between data integrity and data losslessness, as well as between query efficiency and update efficiency, as illustrated in Fig. 3.

Relational normalization appears as a special case of this process where only two c-degrees are available, namely the top and the bottom c-degree. Here, the only FD set we can use to perform classical normalization results from pFDs that have the top c-degree. If more c-degrees are available for a domain, then different requirements of applications in this domain identify different possible worlds and therefore different sets of functional dependencies and tuples they apply to.

Finally, we stress that the instances over the output of the normalization process are classical relations. In particular, the tuples do not carry any p-degrees. By the results of the previous section, the relations do not exhibit any data redundancy nor any form of update anomalies by design. Indeed, the underlying application – according to its requirements – only considers tuples that meet the p-degree threshold, and for such tuples, we do not have data redundancy nor anomalies by design.

7.2. 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 5 motivates the following definition of a BCNF decomposition that is lossless for a given degree of possibility. Again, the definition exploits the relationships between p-degrees associated with tuples of data, and c-degrees associated with FDs.

Definition 17. An α_{k+1-i}-lossless BCNF decomposition of a p-schema $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$ with respect to the pFD set Σ is a lossless BCNF decomposition of R wrt Σ_{β_i}.

Instrumental to Definition 17 is the following decomposition theorem, which follows directly from Theorem 1. It covers the classical decomposition theorem [48] as the special case of having just one possible world.

Theorem 12. *Let $(X \to Y, \beta_i)$ be a pFD with $1 \leq i < k+1$ that satisfies the p-relation $(r, Poss_r)$ over the p-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 projections 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:	Qualitative BCNF Decomposition
INPUT:	Possibilistic Relation Schema $(R, \mathcal{S})$ Set Σ of pFDs over $(R, \mathcal{S})$ Certainty degree $\beta_i \in \mathcal{S}^T - \{\beta_{k+1}\}$
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 on our running example.

Example 18. Let (MEETING, $\mathcal{S}$) and Σ be as in Example 2. As (MEETING, $\mathcal{S}$) is not in β_2-BCNF wrt Σ, we perform an α_3-lossless BCNF decomposition wrt Σ_{β_2}. The result consists of $R_1 = \{Project, Room, Time\}$ with projected FD set

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

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

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

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 the worlds r_1, r_2 and r_3 from Fig. 1, this decomposition may be applied to the world r_3 to obtain the instance $\mathcal{D}_3$ shown in Fig. 2.

The last example is rather special, since one cannot expect to preserve all FDs in the BCNF decomposition process. This is illustrated with another example.

Example 19. Let (MEETING, $\mathcal{S}$) and Σ be as in Example 2. As (MEETING, $\mathcal{S}$) is not in β_4-BCNF wrt Σ, we perform an α_1-lossless BCNF decomposition wrt Σ_{β_4}. The result is $R_1 = \{Manager, Project\}$ with projected FD set

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

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

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

Note that the FD $Manager, Time \to Room$ is not implied by $\Sigma_{\beta_4}[R_1] \cup \Sigma_{\beta_4}[R_2]$. Since the pFDs in Σ apply only to world r_1 from Fig. 1, this decomposition may be applied to r_1 to obtain the instance $\mathcal{D}_1$ shown in Fig. 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 20. A *β-dependency-preserving* decomposition of a p-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 18 is β_2-dependency-preserving, but the α_1-lossless BCNF decomposition from Example 19 is not β_4-dependency-preserving. In fact, for (MEETING, $\mathcal{S}$) and Σ as in Example 2 there is no β_4-dependency-preserving, α_1-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 $Manager, Time \to Room$ after an update, one would have to join R_1 and R_2 from Example 19. This can be prohibitively expensive.

7.3. 3NF synthesis

3NF synthesis guarantees dependency-preservation, but cannot guarantee the elimination of all data value 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 [4,36]. 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 21. A β_i-*dependency-preserving,* α_{k+1-i}-*lossless 3NF decomposition* of a p-schema $(R, \{\alpha_1, \ldots, \alpha_{k+1}\})$ with respect to the pFD set Σ is a dependency-preserving, lossless 3NF decomposition of R wrt Σ_{β_i}.

According to Theorem 12 a β_i-dependency-preserving, α_{k+1-i}-lossless 3NF synthesis with respect to 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:	Qualitative 3NF Synthesis
INPUT:	Possibilistic relation schema $(R, \mathcal{S})$ Set Σ of pFDs over $(R, \mathcal{S})$ Certainty degree $\beta_i \in \mathcal{S}^T - \{\beta_{k+1}\}$
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}

We illustrate the synthesis on our running example.

Example 22. Let (MEETING, $\mathcal{S}$) and Σ be as in Example 2. Since (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 the two schemata $R_1 = \{Project, Room, Time\}$ 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 18.

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 in Section 9.

8. GUI

A web-based GUI[1] provides full access to our algorithms and experiments. It consists of three main parts.

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 [3], the polynomial-time algorithm [53], 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 canonical covers (Option: Unique LHS), or with respect to non-redundant, L-reduced covers in which every FD has only one attribute on the right (Option: Expanded).

Under *Qualitative Normalization*, see Fig. 6, qualitative variants of all of the algorithms above can be applied to a set of pFDs over a p-schema and an optional attribute subset, as specified by a user or randomly generated by the GUI. The user can specify

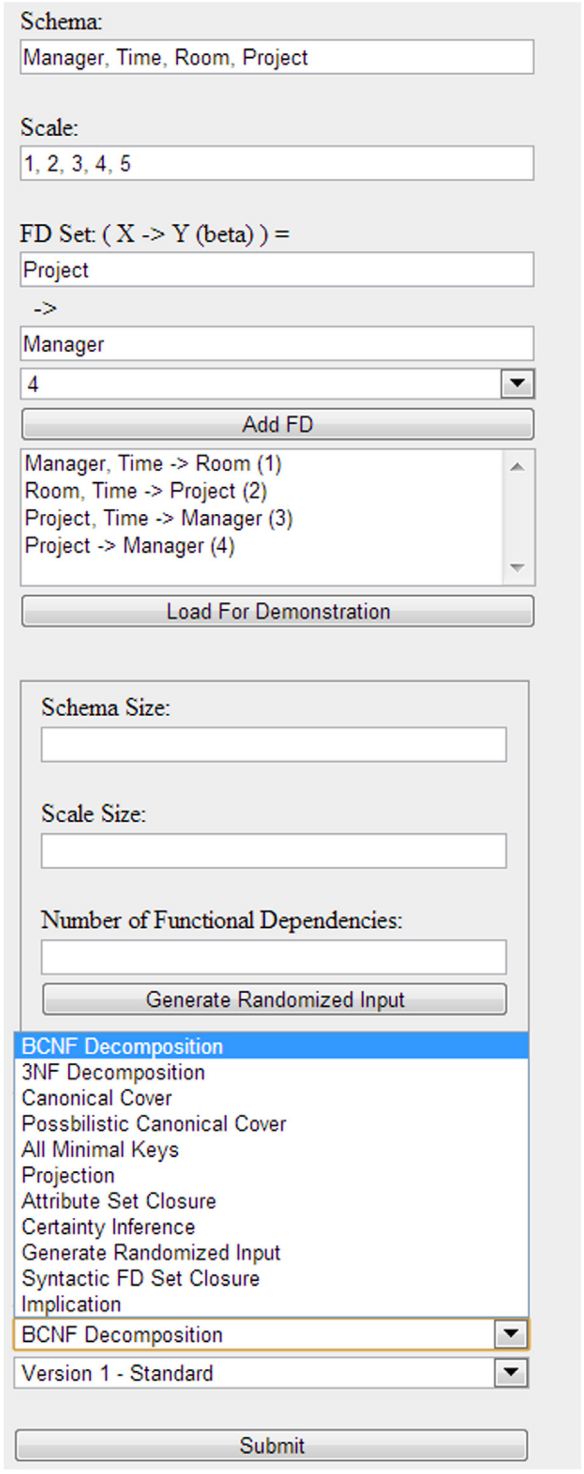

Fig. 6. User interface — normalization.

a scale of c-degrees in the form of positive integers, and assign these to FDs to obtain pFDs. Whenever the user specifies a c-degree (under Scale Index), the GUI applies the scaled version of an algorithm with that c-degree. Otherwise, the same output is returned as for the classical algorithms. As an illustration, Fig. 7 shows the input and output to the β_2-BCNF decomposition from Example 18. The latter figure also shows that the application returns several timings with the output.

[1] http://www.dbschemadesign.cs.auckland.ac.nz/.

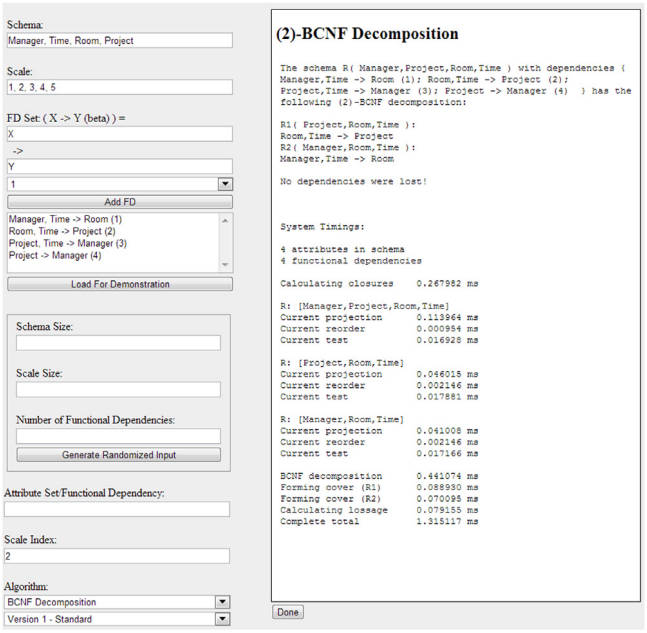

Fig. 7. BCNF example in GUI.

Under *Experiments*, our GUI provides the ability to launch experiments with our algorithms, either via the Web or in a distributed way upon high-performance computing facilities.[2]

9. Experiments

In this section we present and analyze the results of our experiments regarding the qualitative variants of BCNF decompositions and 3NF syntheses. The analysis includes new insights into classical relational schema design trade-offs. We remark that our sole utilization of the existing distributed cluster environment was the parallel execution of our sequential implementations on as many artificially created examples as possible.

Our core analysis concerns the trade-offs between the elimination of data value 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 heuristic measure for the query and update efficiency of a decomposition is the *number of its schemata* - to which we refer as the size of the decomposition. In fact, 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. We measure the size of the decomposition (y-axis) in (1) the given number k of different c-degrees, (2) the given number of different possible worlds, and (3) the size of the given constraint sets. We represent these three measures uniformly as the given maximum c-degree β_k (x-axis). Indeed, the given number k of different c-degrees coincides with the given number of possible worlds, and the size of the input constraint set Σ is that of its β_k-cut. In particular, recall that $\Sigma_{\beta_1} \subseteq \cdots \subseteq \Sigma_{\beta_k}$ holds, so the size of the input constraint sets grows as k grows. Our experiments have been run up to $k = 15$. We argue that this number is representative as decision-making becomes ineffective when an organization must distinguish between higher numbers of degrees for uncertainty. Nevertheless, our framework can also be used to run experiments for larger numbers k.

[2] https://www.nesi.org.nz.

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.

Fig. 8 shows the average size of decompositions and average times to generate them (y-axis) in the given number k of available c-degrees (x-axis: $k = 1, \ldots, 15$). For a fixed k, the averages of β_i-BCNF/3NF decompositions (using the 5 algorithms) were taken over 5000 randomly generated input pFD sets Σ over a fixed relation schema with 15 attributes and the number k of available c-degrees. The maximum number s of pFDs in each set Σ is $s = 15$ in the left column and $s = 75$ in the right column of Fig. 8, respectively.

The experiments show nearly uniform results: The lowest-sized decomposition is BCNF(1), that are 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 FDs and thereby sacrificing the elimination of some data value 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 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 value 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 FDs that were not preserved during the decomposition.

The experiments show that the c-degree is a parameter that can effectively control the size of the decompositions that are produced, and thereby address different levels of query and update efficiency. The graphs illustrate an interesting behavior for the size of the decompositions in the growing size of the input FD set. When there are about as many given FDs as underlying attributes (left column), then the average size of decompositions grows linearly. When there are significantly more given FDs than underlying attributes (that is, 5 times as many FDs in the right column), 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 (which coincide with lower levels of data losslessness) achieve the same levels of query efficiency as significantly lower levels of data integrity (which coincide with higher levels of data losslessness).

The bottom row of Fig. 8 shows the average times to generate the decompositions for our two experiments with $s = 15$ and $s = 75$. BCNF(2) takes the lowest time, followed by BCNF(3), followed by BCNF(1), and followed by 3NF(1,2). These results are intuitive,

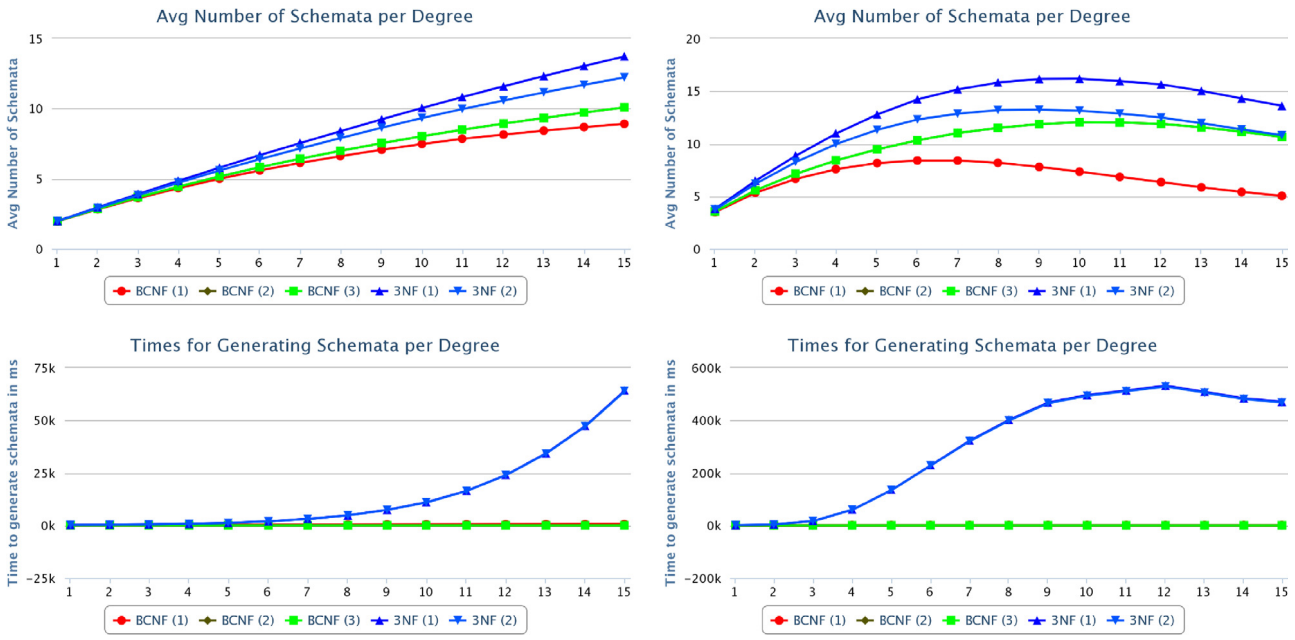

Fig. 8. Average size of decompositions and average times to generate them (y-axis) in the given number k of available c-degrees (x-axis: $k = 1, \ldots, 15$). For a fixed k, the averages were taken over 5000 randomly generated input pFD sets Σ over a fixed relation schema with 15 attributes and the number k of available c-degrees. The maximum number s of pFDs in each Σ is 15 in the left column and 75 in the right column.

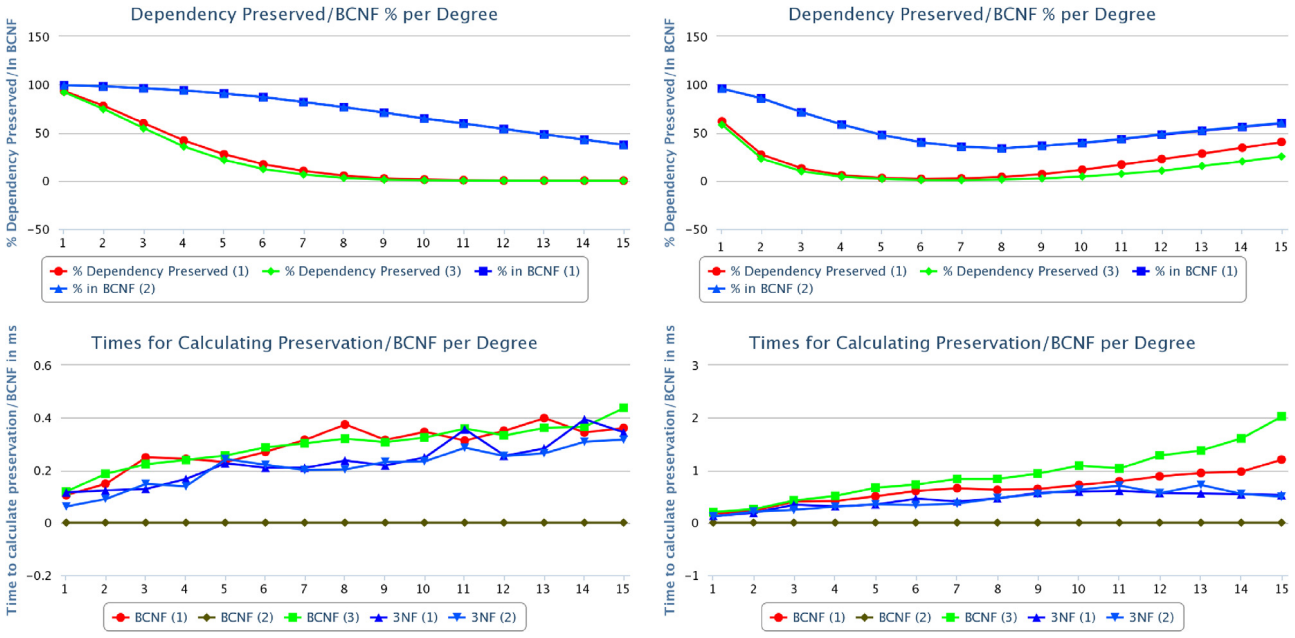

Fig. 9. Percentages of 3NF outputs in BCNF versus BCNF outputs that are dependency-preserving, and average times to check for BCNF in 3NF outputs versus dependency-preservation in BCNF outputs, respectively, for the given number k of available c-degrees (x-axis: $k = 1, \ldots, 15$). For a fixed k, the average was taken over 5000 randomly generated input pFD sets Σ over a fixed relation schema with 15 attributes and the number k of available c-degrees. The maximum number s of pFDs in each Σ is $s = 15$ in the left column and $s = 75$ in the right column.

too. BCNF(2) demonstrates its major strength in comparison to the other algorithms, with a maximum average of 8 ms for $s = 15$ and 13 ms for $s = 75$. BCNF(3) requires slightly more time with a maximum average of 12 ms for $s = 15$ and 35 ms for $s = 75$, resulting from the projections of the given FD sets to the final output schemata at the end. BCNF(1) requires more time with a maximum average of 630 ms for $s = 15$ and 627 ms for $s = 75$, as a result of projecting the given FD sets to all intermediate and final schemata. Finally, 3NF(1,2) requires significantly more time with a maximum average of 64 s for $s = 15$ and 530 s for $s = 75$. Considering the purpose of schema design, all these times can easily be considered as very efficient, and should not be a decisive factor when choosing a final decomposition.

Two main observations illustrate inherent trade-offs. For BCNF decompositions, decompositions of smaller size 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, 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 final decision whether to use a 3NF decomposition that is not a BCNF decomposition or a BCNF decomposition that is not dependency-preserving requires a case by case analysis. Considering update efficiency one must ask whether frequent updates are slowed down more by data value 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 provide better efficiency in evaluating frequent queries.

An important practical question is which of the two normalization strategies is more likely to result in an ultimate decomposition, defined as a lossless, dependency-preserving BCNF decomposition. Therefore, we computed the percentages of those lossless BCNF decompositions that are dependency-preserving and of those lossless 3NF decompositions that are in BCNF, obtained from our randomly created inputs. The results are shown in the top row of Fig. 9. It shows the 3NF strategy as the clear winner. In fact, it is intuitive that the probability of generating some 3NF pattern that violates the BCNF condition is lower than the probability of not preserving some FD. However, the difference is substantial. Indeed, for the case where $s = 15$, the 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 $s = 75$, the 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 strategy in practice, not only in terms of grouping together logically linked attributes but also in terms of the chances for creating an optimal decomposition for which all data value redundancy is eliminated and consistency with respect to all FDs can be enforced locally.

Finally, the bottom row of Fig. 9 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 2 ms.

10. A real-world example from web data extraction

In this section we illustrate our decomposition framework on a real-world example from Web data extraction. Here, basic information about the top-100 best-selling books was extracted from five online book sellers:

1. eBay book store[3]
2. Amazon book store[4]
3. book directory[5]
4. Barnes & Noble[6]
5. Easons.[7]

For our collection the tool Web Scraper[8] was used on a Chrome extension plugin. It extracted the content from the same HTML tags from every book web page. The extracted results had many mistakes initially, since not all books have the same details and the same HTML tags may also have different values. Errors were fixed manually. In total, 344 tuples were extracted over the p-schema (Books, $\{\alpha_1, \ldots, \alpha_5, \alpha_6\}$) with Books = $\{title, author, page, price, publisher\}$. For $i = 1, \ldots, 6$, a tuple was assigned p-degree α_i if and only if it was contained in $6 - i$ of the data sets. The p-degree therefore denotes the possibility by which a book is listed in the top-100 of each book seller. From the resulting p-relation we then computed a canonical cover Σ for the set of pFDs that are satisfied by it. We obtained the following pFD set Σ:

1. $(title \to author, \beta_1)$,
2. $(title, page \to price, \beta_1)$,
3. $(title, publisher \to price, \beta_1)$,
4. $(title, price \to page, \beta_1)$,
5. $(author, price \to publisher, \beta_1)$,
6. $(title \to price, \beta_2)$,
7. $(author, page \to publisher, \beta_2)$,
8. $(author, price \to page, \beta_2)$,
9. $(page, price \to author, \beta_2)$,
10. $(page, publisher \to price, \beta_3)$,
11. $(publisher, price \to page, \beta_3)$,
12. $(page \to price, \beta_4)$,
13. $(publisher \to author, \beta_4)$,
14. $(publisher, price \to title, \beta_4)$,
15. $(publisher \to price, \beta_5)$.

In what follows we list ten different decompositions into β-BCNF and β-3NF together with their properties regarding losslessness, elimination of data redundancy and update anomalies, and preservation of functional dependencies.

- β_1-BCNF, α_5-lossless, no α_5-redundancy, no α_5-update anomalies:
  - $R_1 = \{author, title\}$ with $title \to author$,
  - $R_2 = \{page, price, publisher, title\}$ with $page, title \to price$; $price, title \to page$; $price, title \to publisher$
  - Not β_1-dependency-preserving, e.g. the following two FDs were lost: $publisher, title \to price$; and $author, price \to publisher$
- β_2-BCNF, α_4-lossless, no α_4-redundancy, no α_4-update anomalies:
  - $R_1 = \{author, page, publisher\}$ with $author, page \to publisher$
  - $R_2 = \{author, page, price\}$ with $page, price \to author$ and $author, price \to page$
  - $R_3 = \{page, price, title\}$ with $title \to price$ and $title \to page$
  - β_2-dependency-preserving
- β_3-BCNF, α_3-lossless, no α_3-redundancy, no α_3-update anomalies:
  - $R_1 = \{author, page, price, publisher\}$ with $author, price \to publisher$; $author, page \to price$; $price, publisher \to page$; $page, publisher \to author$; and $page, price \to author$
  - $R_2 = \{author, page, title\}$ with $title \to author$ and $title \to page$
  - β_3-dependency-preserving
- β_4-BCNF, α_2-lossless, no α_2-redundancy, no α_2-update anomalies:
  - $R_1 = \{author, publisher\}$ with $publisher \to author$
  - $R_2 = \{page, price, publisher, title\}$ with $page \to price$; $title \to publisher$; $page \to title$; $price, publisher \to page$; and $title \to page$
  - not β_4-dependency preserving, e.g. the following two FDs were lost: $author, price \to publisher$ and $author, price \to page$

[3] http://www.half.ebay.com/books-bestsellers.
[4] https://www.amazon.com/best-sellers-books-Amazon/zgbs/books.
[5] https://www.bookdepository.com/bestsellers.
[6] http://www.barnesandnoble.com/b/books/_/N-1fZ29Z8q8.
[7] http://www.easons.com/shop/c-bestselling-books.
[8] http://webscraper.io/.

- β_5-BCNF, α_1-lossless, no α_1-redundancy, no α_1-update anomalies:
  - $R = \{author, page, price, publisher, title\}$ with $page \to author$; $title \to page$; $title \to price$; $publisher \to page$; $author, price \to publisher$; and $page \to title$
  - β_5-dependency-preserving
- β_1-3NF, α_5-lossless, β_1-dependency-preserving:
  - Canonical cover:
    - $title \to author$
    - $page, title \to price$
    - $publisher, title \to price$
    - $price, title \to page$
    - $author, price \to publisher$
  - Minimal keys:
    - $\{page, title\}$;
    - $\{price, title\}$;
    - $\{publisher, title\}$
  - 3NF decomposition:
    - $R_1 = \{author, title\}$ with $title \to author$
    - $R_2 = \{page, price, title\}$ with $page, title \to price$ and $price, title \to page$
    - $R_3 = \{price, publisher, title\}$ with $price, title \to publisher$ and $publisher, title \to price$
    - $R_4 = \{author, price, publisher\}$ with $author, price \to publisher$
    - This decomposition satisfies BCNF, so there are no α_5-redundancy and no α_5-update anomalies
- β_2-3NF, α_4-lossless, β_2-dependency-preserving:
  - Canonical cover:
    - $title \to page, price$
    - $author, page \to publisher$;
    - $page, price \to author$;
    - $author, price \to page$
  - Minimal keys:
    - $\{title\}$
  - β_2-3NF decomposition:
    - $R_1 = \{page, price, title\}$ with $title \to price$ and $title \to page$
    - $R_2 = \{author, page, publisher\}$ with $author, page \to publisher$
    - $R_3 = \{author, page, price\}$ with $page, price \to author$ and $author, price \to page$
    - This decomposition satisfies BCNF, so there are no α_4-redundancy and no α_4-update anomalies
- β_3-3NF, α_3-lossless, β_3-dependency-preserving:
  - Canonical cover:
    - $page, publisher \to price$
    - $title \to author, price$
    - $page, price \to author$
    - $author, price \to publisher$
    - $author, page \to publisher$
    - $price, publisher \to page$
  - Minimal keys:
    - $\{title\}$
  - β_3-3NF decomposition:
    - $R_1 = \{page, price, publisher\}$ with $page, price \to publisher$; $page, publisher \to price$; $price, publisher \to page$
    - $R_2 = \{author, price, title\}$ with $title \to price$ and $title \to author$
    - $R_3 = \{author, page, price\}$ with $author, page \to price$; $page, price \to author$; and $author, price \to page$
    - $R_4 = \{author, price, publisher\}$ with $author, price \to publisher$ and $price, publisher \to author$
    - $R_5 = \{author, page, publisher\}$ with $author, page \to publisher$ and $page, publisher \to author$
    - This decomposition satisfies BCNF, so there are no α_3-redundancy and no α_3-update anomalies
- β_4 3NF, α_2-lossless, β_4-dependency-preserving:
  - Canonical cover:
    - $price, publisher \to title$;
    - $author, price \to page$;
    - $title \to page$;
    - $page \to price, publisher$;
    - $publisher \to author$
  - Minimal keys:
    - $\{title\}$,
    - $\{page\}$,
    - $\{author, price\}$,
    - $\{price, publisher\}$
  - β_4-3NF decomposition:
    - $R = \{author, page, price, publisher, title\}$ with $price, publisher \to title$; $author, price \to page$; $title \to page$; $page \to price, publisher$; and $publisher \to author$
    - This schema violates BCNF, so there are examples of α_1-redundancy and α_1-update anomalies
- β_5-3NF, α_1-lossless, β_5-dependency-preserving:
  - Canonical cover:
    - $author, price \to page$
    - $publisher \to title$
    - $page \to author, price, publisher$
    - $title \to page$
  - Minimal keys:
    - $\{title\}$,
    - $\{page\}$,
    - $\{author, price\}$,
    - $\{publisher\}$
  - β_5-3NF decomposition:
    - $R = \{author, page, price, publisher, title\}$ with $author, price \to page$; $publisher \to title$; $page \to author, price, publisher$; and $title \to page$
    - This decomposition satisfies BCNF, so there are no α_1-redundancy and no α_1-update anomalies

We conclude this section with the following remarks. Three out of the five BCNF decompositions are dependency-preserving, while four out the five 3NF decompositions are in BCNF. Furthermore, the BCNF decompositions consisted of two relation schemata on average, while the 3NF decompositions consisted of

almost 3 relation schemata on average. These trends are intuitive and consistent with our experiments from the previous section. In particular, the size of the resulting decompositions illustrates the trade-off between update efficiency (achieved by dependency-preservation) and query efficiency (achieved by decompositions of smaller size, since less joins are required).

Note that all decompositions in our example were purely reliant on the FD discovered from the given data instance. Our goal of this section was not to discuss the meaningfulness of the discovered FDs for the given application domain, but simply to illustrate our decomposition framework and the semantic achievements of the different normal forms. In practice, normalization should only rely on input FDs that are meaningful for the application domain.

11. Related work

We comment on complementary approaches to deal with uncertainty in data, and highlight the novelty of our contributions in the context of previous research.

There are two major and complementary approaches to dealing with uncertainty in data. In general, probability theory is quantitative with more precise outcomes, but these come at the price of acquiring actual probabilities and high computational complexities in managing them [51]. Possibility theory can accommodate qualitative information, and the acquisition and computation of possibilities and certainties is simpler [23]. Consider for example a data integration scenario where different degrees of trust in data sources are easily captured using a possibilistic data model, while a probabilistic approach requires a mapping of trust to precise probability values, and if such a mapping is not meaningful (that is, "made up because we need one"), the accuracy of any derived results is questionable. However, if data uncertainty does come with meaningful probability values, for example from statistical analysis, a probabilistic model is more appropriate if it can be managed with feasible resources. Hence the choice between probabilistic and possibilistic data models should depend on the available uncertainty information and available resources. In recent work, a survey of practical methods for constructing possibility distributions was given [25].

Research on probabilistic databases has focused on queries [51]. Typical is the desire to extend trusted relational technology to handle uncertainty. This is also inherent here: We show how to exploit relational normalization for schema design of applications with qualitatively uncertain data.

Probabilistic databases address the need to deal with uncertain data when meaningful probability distributions are available [51]. Constraints present a key challenge: "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" [16]. 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" [51]. Hence, a qualitative approach to uncertainty, such as the one that possibility theory enables [13,24,45], can avoid the high complexity in obtaining and processing probabilities, while guaranteeing the same qualitative outcome. Here, we refer to the *same* qualitative outcome in the case where only the ranking is presented to the user, but not the degree of uncertainty (e.g. the probability and the p/c-degree, respectively).

As mentioned, our approach differs from quantitative frameworks by its qualitative nature. Moreover, it is beneficial to point out *how* it fundamentally departs from probabilistic modeling. Our p-relations and their possible world semantics may give the impression that they can be modeled as probabilistic conditional (PC) tables [51]: Each tuple t with associated p-degree α_i in a p-relation would be assigned the propositional formula $X = \alpha_1 \vee \cdots \vee X = \alpha_i$ in the PC-table. In fact, the results of our framework depend on the downward closure property of FDs. However, to take advantage of the downward closure property in PC-tables one would have to restrict their use of propositional formulae to the ones above, that is, $X = \alpha_1 \vee \cdots \vee X = \alpha_i$. However, the p-degrees do not obey the rules of the probability calculus. Importantly, there is no research in which constraints on probabilistic databases have been associated with the data or worlds to which they apply. Therefore, probabilistic databases do not have a notion that corresponds to our c-degrees and provides the fundamental control mechanism to define different degrees of data value redundancy and the normal forms that target the elimination/minimization of these, as shown in Fig. 3. Indeed, constraints on probabilistic databases are applied to clean data, in which possible worlds that violate them are simply removed, the probability distribution is adjusted according to the weight of a constraint, or repairs are sampled [51]. We are therefore unaware of any research in which probabilistic constraints have been applied to develop a database schema design theory, in particular a relational one. This, however, is the focus of the current article in the context of a possibilistic model. In fact, the possibilistic framework is characterized by the existence of dual notions of possibility and certainty: something is all the more certain as its negation is less possible, while probabilities are auto-dual, that is, $Prob(\neg X) = 1 - Prob(X)$. Lastly, the logic of possibilistic FDs behaves like possibilistic logic [22], that is, it is sound and complete on the basis of the repeated use of a min-based inference rule, while the probabilistic counterpart, namely $Prob(Y) \geq \max\{0, Prob(X) + Prob(X \rightarrow Y) - 1\}$, would not lead to a complete calculus [21].

Few papers address schema design for uncertain data [13,50]. Das Sarma, Ullman and Widom develop an "FD theory for data models whose basic construct for uncertainty is *alternatives*" [50]. That work is thus fundamentally different from the current approach. In particular, p-relations cannot always be expressed by the uncertain relations of [50]. For example, the simple two-tuple p-relation $\{(t_1, \alpha_1), (t_2, \alpha_2)\}$ with the possible worlds $w_1 = \{t_1\} \subseteq \{t_1, t_2\} = w_2$ cannot be expressed: The world w_1 says there is at most one tuple in which t_1 is an alternative, while the world w_2 says that there must be at least two tuples, namely one tuple in which t_1 is an alternative and one tuple in which t_2 is an alternative. Indeed, t_1 and t_2 cannot be alternatives of the same tuple since possible worlds in [50] result from choosing one alternative from each tuple. The article [13] models fuzziness in an Entity-Relationship model, so addresses schema design by a conceptual approach and not by a logical approach as we do in the current article. [13] derives the uncertainty of their fuzzy functional dependencies from fuzzy similarity relations between attribute values, as proposed in [47]. That means that classical normalization is applied to data value redundancy with weaker notions of value equality, but always to the same set of tuples and the same set of FDs. Instead, the certainty of pFDs in the current article is derived from the largest possible world of tuples to which they apply. That means classical normalization is still applied to data value redundancy based on value equality, but optimized in terms of the number of FDs that apply to a possible world. Both approaches are therefore incomparable.

Possibilistic functional dependencies, as used in the current article, were introduced in [39], where their novelty over other possibilistic notions of FDs is explained, their grounding with the rules of possibility theory is established, and the equivalence of their implication problem with that of Horn clauses in possibilistic propositional logic is proven, capturing Fagin's result from traditional FDs and Boolean Horn clauses as the special case

where $k = 1$ [27]. Reasoning about pFDs and their use for schema design, which is the main practical motivation for this notion of a pFD, are the contributions of the current article. Reasoning about keys and cardinality constraints have also been studied in [31,33] in the context of our possibilistic data model.

The present article is an extension of [40]. Multiple additions were made: the present article contains all proofs, and an extended presentation with more motivation and examples. Update anomalies were not discussed in [40], but provide an important semantic justification for our scaled Boyce–Codd normal forms. Additional experiments were included in the present article, illustrating trade-offs between Boyce–Codd and Third normal forms, and highlighting that synthesizing schemata into Third normal form provides a better chance of obtaining a dependency-preserving, lossless Boyce–Codd normal form than the Boyce–Codd normal form decomposition approach does. Finally, we have also included a real-world possibilistic example, in which tuples about books were extracted from five different online stores, and assigned p-degrees with respect to the number of stores in which they occur. Subsequently, the pFDs that hold on the resulting p-relation were mined, and decompositions into β-BCNF and β-3NFs were applied.

Conditional functional dependencies target the cleaning of certain data and not schema design [30], while pFDs target schema design for uncertain data and not data cleaning. While possibilistic conditional functional dependencies may provide an interesting notion for data cleaning, they are not the focus of the current article and will be studied in the future. Nevertheless, it is useful to compare pFDs with conditional functional dependencies. In fact, pFDs can be expressed by conditional functional dependencies if we view our p-degrees as data on which the conditions for conditional functional dependencies can be defined. For example, to view the pFD $(Manager, Time \rightarrow Room, \beta_1)$ as a conditional functional dependency, the tableau associated with the conditional functional dependency contains a column for the p-degrees with entry $\{\alpha_1, \ldots, \alpha_k\}$, which stipulates to which tuples the FD $Manager, Time \rightarrow Room$ applies. In this sense, the contributions of our article show that pFDs form a fragment of conditional functional dependencies that is particularly relevant to the relational database schema design of uncertain data, similar to how FDs form a fragment of conditional functional dependencies that is particularly relevant to relational database schema design of certain data. In particular, we also show that pFDs enjoy the same efficient computational properties as traditional FDs, while reasoning about conditional functional dependencies is intractable in general [30], and therefore not useful for schema design methods. Another important difference is that conditional functional dependencies are defined on certain data, while we are interested in uncertain data. This difference in target applications is emphasized by our possible world semantics, which is fundamental to our definitions and results, while conditional functional dependencies are only defined on the single possible world which is certain.

Relational schema design for temporal databases with multiple granularities has been studied in [58]. Here, time granularities form a lattice with respect to containment of the time intervals. The authors establish an axiomatization for their notion of temporal functional dependencies, and propose temporal versions of Boyce–Codd and Third normal forms, without semantically justifying them. Basically, data redundancy (without formally defining it) is avoided by rolling out the data with respect to the right time granularity. In contrast, our work deals primarily with uncertainty, which is not a dimension considered in [58]. In particular, no degrees of possibility are assigned to data and no dual degrees of certainty are assigned to functional dependencies. The use of certainty degrees allows us to avoid data redundancy up to the degree that we target. Despite these fundamental differences, the axiomatization of the temporal FDs is syntactically similar to the axiomatization of our possibilistic FDs. It would be interesting future work to investigate schema design for uncertain temporal databases.

Deduction, which is captured by datalog rules, in the presence of contradictions, which are captured by violations of functional dependencies, have recently been studied in [1,2]. A simple nondeterministic semantics was proposed for datalog with FDs based on inferring facts one at a time, never violating the FDs. They also discuss a set-at-a-time semantics, where at each iteration, all facts that can be inferred are added to the database, and then choices are made between contradicting facts. It was shown how to capture possible worlds along with their probabilities via PC-tables. In addition, the problems of computing the probabilities of answers, of identifying most likely supports for answers, and of determining the extensional facts that are most influential for deriving a particular fact were studied. In this sense, the violations of functional dependencies are used to derive probabilities of query answers and their explanations. In our approach we use the downward-closure property of functional dependencies to stipulate the qualitative degree of certainty by which they hold. The degrees of certainty for functional dependencies allow us to develop a schema design methodology for applications with uncertain data.

Approximate dependencies, e.g. [41], tolerate violations of the given constraints up to a given threshold. Our constraints are not approximate: They are either satisfied or violated by the tuples to which they apply. It is future work to investigate approximate versions of our constraints.

Horizontal decompositions have been studied for the relational model of data to accommodate exceptions to functional dependencies that should hold on the given schema [44]. The main aim is to partition a relation horizontally into the part of the relation that satisfies a given FD and the part of the relation that has exceptions to the satisfaction of the FD. In our framework, we partition the largest possible world r_k into the largest possible world $r_i \subseteq r_k$ that meets the minimum p-degree requirement and the complement $r_k - r_i$. In particular, we regard the complement as unworthy for the application. It would make sense to combine both approaches in future research. This would accommodate the handling of exceptions in the possibilistic model, allowing horizontal decompositions for each of the possible worlds.

Similar to how our model assigns p-degrees to data and c-degrees to functional dependencies, previous work on multilevel relations assign authorization labels to data [43]. That work is primarily concerned with restricting access for users to those parts of the data which they are authorized to see. An important aspect is the guarantee of entity and referential integrity, as enforced by keys and foreign keys, respectively. The model is based on assigning sensitivity labels at the attribute level. Our model is primarily concerned with schema design for uncertain data, based on different degrees of data redundancy caused by functional dependencies. It is based on assigning possibility degrees on the tuple level. In the future it would be interesting to develop a normalization framework that is based on uncertainty at the attribute level.

Many classical results enabled us to develop schema design for uncertain data. FDs are already mentioned in the seminal paper by Codd [14] and constitute one of the most prolific concepts in databases. Armstrong axiomatized FDs [5], and linear-time algorithms to decide their implication problem are known [6, 18]. These results provide the foundation for relational schema design, including 3NF [7,10,12], BCNF [8,15] and their semantic justification as to dependency-preservation and minimization of data redundancy [4,36,55]. Deciding if a schema is in 3NF is

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

	α_{k+1-i}-lossless	No α_{k+1-i}-redundancy	No α_{k+1-i}-update anomaly	β_i-dependency preserving	Computational complexity of deciding normal form criteria
β_i-BCNF	✔	✔	✔		P (on schema)/coNP-complete (on projected schema)
β_i-3NF	✔			✔	NP-complete

NP-complete, while the same problem for BCNF is in P-time but deciding if a projection of a schema is in BCNF is coNP-complete [6]. Algorithms that compute a lossless BCNF decomposition in P-time exist [53,59,60]. Our experiments reveal new insight into the classical trade-off between 3NF synthesis and BCNF decomposition.

Schema design is important for any data model, semantic [11], nested [49,56], object-oriented [52], temporal [32], XML [4,36,57], and data models with tuple alternatives [50].

12. Conclusion and future work

We reflect on what we have achieved and how these achievements impact on current database design practice. Finally, we outline future work.

12.1. Conclusion

We have established a framework in which quality relational database schemata are designed for applications that require data of sufficient confidence. Our framework exploits requirements about the degrees of certainty required for functional dependencies that are relevant for a given application, and dually about the degrees of possibility required for tuples considered to be relevant. Indeed, these requirements are exploited to design better relational database schemata.

In technical terms, we have established foundations for a design theory by characterizing the implication problem for functional dependencies with certainty degrees both axiomatically and in terms of a linear time decision algorithm. These foundations enabled us to introduce different degrees of the syntactic Boyce–Codd normal form and to justify them semantically in terms of characterizing schemata whose instances are free from data value redundancy and update anomalies up to the degree that an application requires. Similarly, different degrees of the syntactic Third normal form are introduced and justified semantically in terms of characterizing schemata whose instances exhibit minimal amounts of data value redundancy among all schemata that preserve all given FDs up to the degree that an application requires. Therefore, our variants of the Boyce–Codd and Third normal forms extend their classical properties in terms of losslessness, data redundancy, update anomalies, dependency-preservation and computational complexity, down to any degree of certainty required, as shown in Table 4. In particular, the core of the computational problems rest in deciding β-BCNF for projections of the given schema (coNP-completeness), and deciding β-3NF for a given schema (NP-completeness). Our experiments confirm the effectiveness and efficiency of our framework, and provide original insight into relational schema design trade-offs in terms of query and update performance.

In practical terms, our framework is able to refine the classical database design process by effectively utilizing the degree of certainty that an application requires from the functional dependencies that are relevant for it. In fact, by fixing a degree of certainty, classical decomposition and synthesis approaches can be applied to the set of classical functional dependencies that meet the certainty degree. By design, the outputs of the normalization process are guaranteed to enjoy the properties we have established for this degree of certainty. In other words, by selecting different degrees of certainty, organizations can use our framework to explore different schema design options and make a decision informed by the properties of the different design options and the requirements of the application in terms of the target levels for data integrity, data losslessness, query efficiency and update efficiency, as illustrated in Fig. 3. If degrees of certainty express preference, then our framework can also be seen as unifying relational normalization and de-normalization. In fact, classically de-normalized schemata are fully normalized for the FDs that meet the targeted preference threshold. Our experiments empirically justify the dominant use of 3NF synthesis in practice — showing that it offers better chances of obtaining an optimal schema than BCNF decompositions do.

12.2. Future work

Our current article has addressed the problem how the availability of c-degrees can improve the design of classical relational database schema. A related problem, not addressed in the current article, is the problem of *possibilistic database design* where the result is a possibilistic schema that carries possibilistic instances. In that case we would have p-degrees attached to the tuples, and tuples as well as their p-degrees could then be updated. Along the same lines of research, we may consider possibilistic models of uncertainty with various levels of expressivity, such as possibilistic conditional tables recently introduced in [46].

There are several other areas that also warrant future research. There are all practically motivated by the potential need of applications. For example, partial instead of linear orders of possibility and certainty degrees might be available. Theorem 1, which enables all our results, does no longer hold for partial orders. Linear orders represent the practically most interesting case and form the qualitative counterpart to the linear order of probabilities. We remark that any results in our current article still apply to every chain of any given partial order. Furthermore, degrees of possibility might be available at the attribute instead of the tuple level. Hence, a normalization framework should be developed for this type of information. Applications suffer from incomplete information, for example in the form of missing data values. Hence, another extension of our work to handling null marker occurrences is desirable. Recently, schema normalization in the presence of null markers has been considered as well [34,35]. Just like schema designs for certain data depend on the identification of semantically meaningful FDs, schema designs for uncertain data depend on the identification of semantically meaningful pFDs. As already indicated in Section 7 this strongly motivates the extension of research about Armstrong relations [29] to our qualitative setting. Data redundancy and update anomalies are not only caused by functional dependencies, but other kinds of data dependencies. For certain data, the normalization framework encompasses other normal forms such as 4NF [28,55], essential tuple normal form [17], 5NF [54] and Inclusion Dependency normal form [38]. Hence, it is interesting to extend these normal forms to uncertain data. This, however, is not straightforward as neither multivalued dependencies, nor join dependencies, nor inclusion dependencies are closed downward. Finally, different applications may need to handle data in different formats. Consequently, an extension of our methods to other data models, including XML, RDF, and graph data, would also be interesting.

References

[1] S. Abiteboul, M. Bienvenu, D. Deutch, Deduction in the presence of distribution and contradictions, in: WebDB, 2012, pp. 31–36.
[2] S. Abiteboul, D. Deutch, V. Vianu, Deduction with contradictions in datalog, in: ICDT, 2014, pp. 143–154.
[3] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[4] M. Arenas, Normalization theory for XML, SIGMOD Rec. 35 (4) (2006) 57–64.
[5] W.W. Armstrong, Dependency structures of data base relationships, in: IFIP Congress, 1974, pp. 580–583.
[6] C. Beeri, P.A. Bernstein, Computational problems related to the design of normal form relational schemas, ACM TODS 4 (1) (1979) 30–59.
[7] P.A. Bernstein, Synthesizing third normal form relations from functional dependencies, ACM TODS 1 (4) (1976) 277–298.
[8] J. Biskup, Boyce-Codd normal form and object normal forms, Inf. Process. Lett. 32 (1) (1989) 29–33.
[9] J. Biskup, Achievements of relational database schema design theory revisited, in: Semantics in Databases, 1995, pp. 29–54.
[10] J. Biskup, U. Dayal, P.A. Bernstein, Synthesizing independent database schemas, in: SIGMOD, 1979, pp. 143–151.
[11] J. Biskup, R. Menzel, T. Polle, Y. Sagiv, Decomposition of relationships through pivoting, in: B. Thalheim (Ed.), ER, in: Lecture Notes in Computer Science, vol. 1157, Springer, 1996, pp. 28–41.
[12] J. Biskup, R. Meyer, Design of relational database schemes by deleting attributes in the canonical decomposition, J. Comput. System Sci. 35 (1) (1987) 1–22.
[13] N.A. Chaudhry, J.R. Moyne, E.A. Rundensteiner, An extended database design methodology for uncertain data management, Inform. Sci. 121 (1–2) (1999) 83–112.
[14] E.F. Codd, A relational model of data for large shared data banks, Commun. ACM 13 (6) (1970) 377–387.
[15] E.F. Codd, Further normalization of the database relational model, in: Courant Computer Science Symposia 6: Data Base Systems, 1972, pp. 33–64.
[16] N.N. Dalvi, D. Suciu, Management of probabilistic data: foundations and challenges, in: PODS, 2007, pp. 1–12.
[17] H. Darwen, C.J. Date, R. Fagin, A normal form for preventing redundant tuples in relational databases, in: A. Deutsch (Ed.), 15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012, ACM, 2012, pp. 114–126.
[18] J. Diederich, J. Milton, New methods and fast algorithms for database normalization, ACM TODS 13 (3) (1988) 339–365.
[19] S. Doherty, The Future of Enterprise Data, http://insights.wired.com/profiles/blogs/the-future-of-enterprise-data#axzz2owCB8FFn, 2013.
[20] X.L. Dong, E. Gabrilovich, K. Murphy, V. Dang, W. Horn, C. Lugaresi, S. Sun, W. Zhang, Knowledge-based trust: Estimating the trustworthiness of web sources, PVLDB 8 (9) (2015) 938–949.
[21] D. Dubois, J. Lang, H. Prade, Automated reasoning using possibilistic logic: semantics, belief revision and variable certainty weights, IEEE Trans. Data Knowl. Eng. 6 (1) (1994) 64–71.
[22] D. Dubois, J. Lang, H. Prade, Possibilistic logic, in: D.M. Gabbay, C.J. Hogger, J.A. Robinson, D. Nute (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3, Oxford University Press, 1994, pp. 439–513.
[23] D. Dubois, H. Prade, Fuzzy set and possibility theory-based methods in artificial intelligence, Artificial Intelligence 148 (1–2) (2003) 1–9.
[24] D. Dubois, H. Prade, Possibility theory, in: R.A. Meyers (Ed.), Computational Complexity: Theory, Techniques, and Applications, Springer New York, 2012, pp. 2240–2252.
[25] D. Dubois, H. Prade, Practical methods for constructing possibility distributions, Int. J. Intell. Syst. 31 (3) (2016) 215–239.
[26] R. Fagin, The decomposition versus synthetic approach to relational database design, in: VLDB 1977, 1977, pp. 441–446.
[27] R. Fagin, Functional dependencies in a relational data base and propositional logic, IBM J. Res. Dev. 21 (6) (1977) 543–544.
[28] R. Fagin, Multivalued dependencies and a new normal form for relational databases, ACM TODS 2 (3) (1977) 262–278.
[29] R. Fagin, Horn clauses and database dependencies, J. ACM 29 (4) (1982) 952–985.
[30] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, ACM TODS 33 (2) (2008).
[31] N. Hall, H. Köhler, S. Link, H. Prade, X. Zhou, Cardinality constraints on qualitatively uncertain data, Data Knowl. Eng. 99 (2015) 126–150.
[32] C.S. Jensen, R.T. Snodgrass, M.D. Soo, Extending existing dependency theory to temporal databases, IEEE TKDE 8 (4) (1996) 563–582.
[33] H. Köhler, U. Leck, S. Link, H. Prade, Logical foundations of possibilistic keys, in: JELIA 2014, 2014, pp. 181–195.
[34] H. Köhler, S. Link, SQL Schema design: Foundations, normal forms, and normalization, in: SIGMOD, 2016, pp. 267–279.
[35] H. Köhler, S. Link, SQL Schema design: foundations, normal forms, and normalization, Inf. Syst. 76 (2018) 88–113.
[36] S. Kolahi, Dependency-preserving normalization of relational and XML data, J. Comput. System Sci. 73 (4) (2007) 636–647.
[37] W.-D. Langeveldt, S. Link, Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies, Inf. Syst. 35 (3) (2010) 352–374.
[38] M. Levene, M.W. Vincent, Justification for inclusion dependency normal form, IEEE TKDE 12 (2) (2000) 281–291.
[39] S. Link, H. Prade, Possibilistic functional dependencies and their relationship to possibility theory, IEEE Trans. Fuzzy Syst. 24 (3) (2016) 757–763.
[40] S. Link, H. Prade, Relational database schema design for uncertain data, in: CIKM 2016, 2016, pp. 1211–1220.
[41] J. Liu, J. Li, C. Liu, Y. Chen, Discover dependencies from data - A review, IEEE TKDE 24 (2) (2012) 251–264.
[42] C.L. Lucchesi, S.L. Osborn, Candidate keys for relations, J. Comput. System Sci. 17 (2) (1978) 270–279.
[43] T.F. Lunt, D.E. Denning, R.R. Schell, M.R. Heckman, W.R. Shockley, The seaview security model, IEEE Trans. Softw. Eng. 16 (6) (1990) 593–607.
[44] J. Paredaens, P. De Bra, M. Gyssens, D. Van Gucht, The Structure of the Relational Database Model, Springer, 1989.
[45] O. Pivert, H. Prade, A certainty-based model for uncertain databases, IEEE T. Fuzzy Syst. 23 (4) (2015) 1181–1196.
[46] O. Pivert, H. Prade, Possibilistic conditional tables, in: FoIKS, in: Lecture Notes in Computer Science, vol. 9616, 2016, pp. 42–61.
[47] K. Raju, A.K. Majumdar, Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems, ACM Trans. Database Syst. 13 (2) (1988) 129–166.
[48] J. Rissanen, Independent components of relations, ACM TODS 2 (4) (1977) 317–325.
[49] M.A. Roth, H.F. Korth, The design of ¬1nf relational databases into nested normal form, in: SIGMOD, 1987, pp. 143–159.
[50] A.D. Sarma, J.D. Ullman, J. Widom, Schema design for uncertain databases, in: AMW, 2009.
[51] D. Suciu, D. Olteanu, C. Ré, C. Koch, Probabilistic Databases, Morgan & Claypool Publishers, 2011.
[52] Z. Tari, J. Stokes, S. Spaccapietra, Object normal forms and dependency constraints for object-oriented schemata, ACM TODS 22 (4) (1997) 513–569.
[53] D.-M. Tsou, P.C. Fischer, Decomposition of a relation scheme into Boyce-Codd normal form, SIGACT News 14 (3) (1982) 23–29.
[54] M.W. Vincent, A corrected 5NF definition for relational database design, Theoret. Comput. Sci. 185 (2) (1997) 379–391.
[55] M.W. Vincent, Semantic foundations of 4NF in relational database design, Acta Inf. 36 (3) (1999) 173–213.
[56] M.W. Vincent, M. Levene, Restructuring partitioned normal form relations without information loss, SIAM J. Comput. 29 (5) (2000) 1550–1567.
[57] M.W. Vincent, J. Liu, C. Liu, Strong functional dependencies and their application to normal forms in XML, ACM Trans. Database Syst. 29 (3) (2004) 445–462.
[58] X. Wang, C. Bettini, A. Brodsky, S. Jajodia, Logical design for temporal databases with multiple granularities, ACM Trans. Database Syst. 22 (2) (1997) 115–170.
[59] Y. Zhang, M.E. Orlowska, An improvement on the automatic tool for relational database design, Inf. Syst. 15 (6) (1990) 647–651.
[60] Y. Zhang, M.E. Orlowska, A new polynomial time algorithm for BCNF relational database design, Inf. Syst. 17 (2) (1992) 185–193.