

Normalisation in the Presence of Lists

Sebastian Link and Sven Hartmann

Information Science Research Centre
Massey University,
Private Bag 11222, Palmerston North, New Zealand,
Email: [S.Link|S.Hartmann]@massey.ac.nz

Abstract

In the relational data model (RDM), normal forms are conditions for relation schemata that a database design should satisfy to ensure an absence of processing difficulties with the database. One prime example of such a normal form is the Boyce-Codd normal form guaranteeing the absence of redundancies and update anomalies caused by functional dependencies (FDs).

Many different data models have been introduced over the years trying to capture data beyond relational structures. The success of those advanced data models will depend, in particular, on the study of normal forms. In fact, finding a unifying framework and extending achievements of relational databases to deal with advanced database features such as complex object types are currently two major challenges in database design (Biskup 1995, Biskup 1998).

Such a unifying approach, capturing various different data models at a time, can be based on the type systems underlying the various data models. In the present paper, we study normalisation in the presence of base, record and finite list types. Nested lists are used as a data structure whenever order matters. List types are therefore supported by many advanced data models such as genomic sequence, deductive and object-oriented data models including XML.

On the basis of a finite axiomatisation of FDs in the presence of lists the Nested List Normal Form (NLNF) is proposed as a weaker normal form than BCNF. This proposal is semantically justified by formally proving that NLNF is equivalent to the absence of redundancy. Moreover, NLNF is equivalent to the absence of strong insertion and most forms of replacement anomalies, and sufficient for the absence of all types of update anomalies.

Keywords: Advanced Data Models, Database Design, Normalisation, Functional Dependencies, Lists

1 Introduction

1.1 Normalisation in Relational Databases

Relational database systems have evolved to an industry standard since their invention by E.F. Codd in 1970 (Codd 1970). This is a result of the simplicity and the sound theoretical basis of the relational data model (RDM). One important issue associated with the use of relational databases is the correct structure or design of data to be used. Several criteria, referred to as *normal forms*, have been proposed as conditions for relation schemata that a database design should satisfy to ensure an absence of processing difficulties with the database. These normal forms give a database designer unambiguous guidelines in deciding which databases are good in the quest to avoid

bad designs that have redundancy problems and update anomalies. Such normal forms have already been introduced in (Codd 1972) by Codd himself. In general, they are dependent on the type of integrity constraints or rules which apply to data items within the database. The most important class of integrity constraints are *functional dependencies* (FDs) which have been intensely studied since their introduction in (Codd 1972). FDs cause difficulties such as redundancy in the representation of data and update anomalies. Codd proposed the Boyce-Codd normal form (BCNF) in (Codd 1974) to overcome these difficulties. Intuitively, a relation schema is in BCNF if it is in first normal form and the left-hand side of each nontrivial FD in the given set of FDs is a superkey for the relation schema with respect to this set of FDs. Codd conjectured that BCNF is an exact condition on a relation schema that avoids redundancies and update anomalies. Later on, after the notions of redundancy and update anomaly had been clarified and formalised, it was shown in (Bernstein & Goodman 1980), (Fagin 1981) and (Vincent 1994) that this is indeed the case. Thus, BCNF is a completely justified normal form in that sense. This is a big step towards automatically verifying whether a relation schema is well-designed. Since BCNF is a simple syntactic condition, the results above show that the relation schema is indeed well-designed in the sense that no redundancies and no update anomalies in terms of FDs can occur.

1.2 Challenges with Advanced Data Models

Nowadays there is a need to store data beyond relational structure, and many different advanced data models have been proposed over the last years. First, so called semantic data models have been developed (Chen 1976), (Hull & King 1987), which were originally just meant to be used as design aids, as application semantics was assumed to be easier captured by these models (Batini & Ceri & Navathe 1992), (Chen 1983), (Tjoa & Berger 1993). Later on some of these models, especially the nested relational model (Paredaens & De Bra & Gyssens & Van Gucht 1989), object oriented models (Schewe & Thalheim 1993) and object-relational models, the gist of which are captured by the higher-order Entity-Relationship model (HERM, (Thalheim 1992, Thalheim 2000)) have become interesting as data models in their own right. Most recently, the major research interest is on the model of semi-structured data and XML (Abiteboul & Buneman & Suciu 2000), which may also be regarded as some kind of object oriented model.

One key problem is to develop dependency theories (or preferably a unified theory) for the most relevant advanced data models. These are probably the HERM as a nested model with various bulk

type constructors, good theoretical foundations and proven practical relevance (Thalheim 2000), the object oriented model (Schewe & Thalheim 1993), the semi-structured data model and XML (Abiteboul & Buneman & Suciú 2000), which add unions and most importantly references, the expansion of which leads to rational tree structures. The development of such a dependency theory will have a significant impact on understanding application semantics and laying the grounds for a logically founded theory of well-designed databases.

Biskup (Biskup 1995, Biskup 1998) lists in particular two challenges for database design theory: finding a unifying framework and extending achievements to deal with advanced database features such as complex object types.

1.3 Contributions

This paper continues to take on these two challenges. In order to provide a unifying framework and to capture several data models at a time our work is based on an abstract algebraic approach in the context of nested attributes. Recall that the set of all subsets of a relation schema forms a Boolean Algebra with respect to inclusion and the set operations of union, intersection and difference. This simple algebraic foundation is the basis for the success of relational database design. In order to extend these achievements to complex object databases we will follow a similar algebraic approach.

In order to show how this approach works with complex object types we consider the finite list type first. This choice has several reasons. The finite list type is perhaps the most common data type. The need for lists arises from applications that store ordered relations, time-series data, meteorological and astronomical data streams, runs of experimental data, multidimensional arrays, textual information, voices, sound, images, video, etc. Lists have been subject to studies in the deductive and temporal database community for some time (Richardson 1992), (Naqvi & Tsur 1989). The list type also naturally appears in object-oriented databases (Schewe & Thalheim 1993), (Gardarin & Cheiney & Kiernan & Pastre & Stora 1989) and is in particular important for XML (Abiteboul & Buneman & Suciú 2000), (W3C 2001). Recently, bioinformatics has become a very important field of research. Of course, lists occur naturally in genomic sequence databases (Seshadri & Livny & Ramakrishnan 1996), (Bry & Kröger 2003).

A finite axiomatisation of FDs in the presence of lists has been provided in (Hartmann & Link & Schewe 2004). This axiomatisation is very close to Armstrong's original set of axioms for the implication of FDs in relational databases (Armstrong 1974). That is, reflexivity, augmentation and transitivity are the three defining properties sufficient for capturing implication of FDs even in the presence of lists. It will turn out in this paper that the simplicity of this axiomatisation allows for a natural generalisation of normalisation. Note that the simplicity and elegance of relational database design are key reasons for the success of the RDM. Finding an elegant and natural generalisation to complex object types within a unifying framework is therefore a big step forward.

The first contribution of this paper is the formal definition of processing difficulties in the presence of lists. It turns out that the notion of redundancy cannot simply be extended to our new framework, but needs some adaptation. Moreover, we will formally define various update anomalies caused by FDs in the presence of lists.

The second contribution is to provide syntactical conditions that describe well-designed nested at-

tributes in the presence of lists with respect to a given set of FDs. The article proposes the Nested List Normal Form (NLNF) which turns out to be strictly weaker than a simple generalisation of BCNF. Another contribution is the characterisation of NLNF. This characterisation makes it possible to automatically check whether a given nested attribute is indeed in NLNF with respect to a given set of FDs.

The most important contribution is the semantic justification of this new normal form. It is demonstrated that the integrity of nested attributes in NLNF can be tested more efficiently. The paper demonstrates formally that nested attributes which are in NLNF with respect to a given set of FDs are well-designed. That is, nested attributes are in NLNF with respect to a given set of FDs if and only if this nested attribute is non-redundant with respect to the given FDs. A further semantic justification is provided by the facts that NLNF is equivalent to the absence of insertion anomalies, and equivalent to the absence of two forms of replacement anomalies. Furthermore, NLNF is sufficient for the absence of another type of replacement anomaly. These results extend results from relational database design.

Finally, the paper shows that the algebraic approach based on nested attributes seems to be very natural. That is, established results from relational database theory can be carried over to complex object databases. The nesting of attributes can be easily extended to sets, multisets, unions, references etc. Therefore, our work might serve as a unifying framework for the foundation of dependency theory in advanced data models.

1.4 Outline

The article is structured as follows. First, we discuss briefly some related work in Section 2. Subsequently, the abstract data model based on typing and nested attributes is introduced in Section 3. The definition of FDs and their finite axiomatisation from (Hartmann & Link & Schewe 2004) are repeated in Section 4. The issue of redundancies in the presence of lists is treated in Section 5. As in the RDM, redundancy can be defined in terms of the set of FDs that is given by the designer or in terms of all FDs implied by such a given set. It is shown that both notions coincide. The Nested List Normal Form (NLNF) is proposed as a normal form for nested attributes. The first main result is that NLNF is equivalent to the absence of redundancy caused by FDs. NLNF is strictly weaker than Boyce-Codd normal form. Furthermore, NLNF is characterised in two different ways making it practically relevant and giving some more insight. In Section 6 the dynamic property of abnormal update behaviour is studied. Strong insertion anomalies and three different types of replacement anomalies are considered. It is proven that NLNF is equivalent to the absence of strong insertion anomalies and equivalent to the first two types of strong replacement anomalies. Moreover, NLNF is sufficient for the absence of the third type of replacement anomalies. We conclude in Section 7 and comment on future work in Section 8.

2 Related Work

Dependency and normalisation theory have been studied in the context of advanced data models. The works in (Mok & Ng & Embley 1996) and (Özsoyoglu & Yuan 1987) consider the nested relational data model where the nesting refers to sets rather than lists. Both articles define functional dependencies

based on the notion of a path, and deviate significantly from our approach.

The work in (Tari & Stokes & Spaccapietra 1997) addresses the development of a normalisation theory for object-oriented data models that have common features to support objects. They provide an extension of functional dependencies to cope with the richer semantics of relationships between objects, called path dependency, local dependency, and global dependency constraints. An object is said to be in normal form if and only if the user's interpretation of this object is derivable from the model of the object. Again, this approach is very different from ours.

The papers (Arenas & Libkin 2002) and (Vincent 2003) study the problem of normalisation in the context of XML. In both articles, the definition of FDs is based on the notion of a path. There are, however, different types of FDs which lead to a different expressiveness. See (Hartmann & Link 2003) for a discussion. Normalisation should then be studied from different points of view.

The approach in this paper specifically focuses on lists and is of algebraic nature. The authors are not aware of any similar work in the literature.

3 The Algebra of Nested Attributes

This paper considers record and finite list types. Nested attributes, however, can be defined with respect to many more types such as sets, multisets, unions and references.

3.1 Nested Attributes

We start with the definition of flat attributes and values for them.

Definition 3.1. A *universe* is a finite set \mathcal{U} together with domains (, i.e., sets of values) $dom(A)$ for all $A \in \mathcal{U}$. The elements of \mathcal{U} are called *flat attributes*. \square

For the relational data model a universe was sufficient. That is, a relation schema is defined by a finite and non-empty subset $\mathcal{R} \subseteq \mathcal{U}$. For higher-order data models, however, nested attributes are needed. In the following definition we use a set \mathcal{L} of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Definition 3.2. Let \mathcal{U} be a universe and \mathcal{L} a set of labels. The set $\mathcal{NA} = \mathcal{NA}(\mathcal{U}, \mathcal{L})$ of *nested attributes over \mathcal{U} and \mathcal{L}* is the smallest set satisfying the following conditions:

- $\lambda \in \mathcal{NA}$,
- $\mathcal{U} \subseteq \mathcal{NA}$,
- for $L \in \mathcal{L}$ and $N_1, \dots, N_k \in \mathcal{NA}$ with $k \geq 1$ we have $L(N_1, \dots, N_k) \in \mathcal{NA}$,
- for $L \in \mathcal{L}$ and $N \in \mathcal{NA}$ we have $L[N] \in \mathcal{NA}$.

We call λ *null attribute*, $L(N_1, \dots, N_k)$ *record-valued attribute* and $L[N]$ *list-valued attribute*. \square

We can now extend the mapping dom from flat attributes to nested attributes, i.e., we define a set $dom(N)$ of values for every nested attribute $N \in \mathcal{NA}$.

Definition 3.3. For a nested attribute $N \in \mathcal{NA}$ we define the *domain* $dom(N)$ as follows:

- $dom(\lambda) = \{ok\}$,

- $dom(L(N_1, \dots, N_k)) = \{(v_1, \dots, v_k) \mid v_i \in dom(N_i) \text{ for } i = 1, \dots, k\}$, i.e., the set of all k -tuples (v_1, \dots, v_k) with $v_i \in dom(N_i)$ for all $i = 1, \dots, k$, and
- $dom(L[N]) = \{[v_1, \dots, v_n] \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\}$, i.e., the set of all finite lists with elements in $dom(N)$. \square

The empty list is denoted by $[\]$. For example, the domain of $L[\lambda]$ is the set of all finite lists where all entries in the list are *ok*, i.e., $Dom(L[\lambda]) = \{[\], [ok], [ok, ok], \dots\}$.

Note that the relational data model is completely covered by the presence of tuple-valued attributes only. Instead of relation schemata R we will now consider a nested attribute N , assuming that a universe \mathcal{U} and a set \mathcal{L} of labels are fixed. An R -relation r is then replaced by some finite set $r \subseteq dom(N)$.

3.2 Subattributes

Dependency theory in the relational data model is based on the powerset $\mathcal{P}(R)$ for a relation schema R . In fact, $\mathcal{P}(R)$ is a powerset algebra with partial order \subseteq , set union \cup , set intersection \cap and set difference $-$. We will generalise these operations for nested attributes starting with a partial order \leq .

Definition 3.4. The *subattribute relation* \leq on the set of nested attributes \mathcal{NA} over \mathcal{U} and \mathcal{L} is defined by the following rules, and the following rules only:

- $N \leq N$ for all nested attributes $N \in \mathcal{NA}$,
- $\lambda \leq A$ for all flat attributes $A \in \mathcal{U}$,
- $\lambda \leq N$ for all list-valued attributes $N \in \mathcal{NA}$,
- $L(N_1, \dots, N_k) \leq L(M_1, \dots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \dots, k$,
- $L[N] \leq L[M]$ whenever $N \leq M$.

For $N, M \in \mathcal{NA}$ we say that M is a *subattribute* of N if and only if $M \leq N$ holds. We write $M \not\leq N$ if and only if M is not a subattribute of N . \square

The subattribute relation \leq on nested attributes is reflexive, anti-symmetric and transitive.

Lemma 3.1. *The subattribute relation is a partial order on nested attributes.* \square

Informally, $M \leq N$ for $N, M \in \mathcal{NA}$ if and only if M comprises at most as much information as N does. The informal description of the subattribute relation is formally documented by the existence of a projection function $\pi_M^N : Dom(N) \rightarrow Dom(M)$ in case $M \leq N$ holds.

Definition 3.5. Let $N, M \in \mathcal{NA}$ with $M \leq N$. The *projection function* $\pi_M^N : Dom(N) \rightarrow Dom(M)$ is defined as follows:

- if $N = M$, then $\pi_M^N = id_{Dom(N)}$ is the identity on $dom(N)$,
- if $M = \lambda$, then $\pi_\lambda^N : Dom(N) \rightarrow \{ok\}$ is the constant function that maps every $v \in Dom(N)$ to *ok*,
- if $N = L(N_1, \dots, N_k)$ and $M = L(M_1, \dots, M_k)$, then $\pi_M^N = \pi_{M_1}^{N_1} \times \dots \times \pi_{M_k}^{N_k}$ which maps every tuple $(v_1, \dots, v_k) \in Dom(N)$ to $(\pi_{M_1}^{N_1}(v_1), \dots, \pi_{M_k}^{N_k}(v_k)) \in Dom(M)$, and

- if $N = L[N']$ and $M = L[M']$, then $\pi_M^N : \text{Dom}(N) \rightarrow \text{Dom}(M)$ maps every list $[v_1, \dots, v_n] \in \text{Dom}(N)$ to the list $[\pi_{M'}^{N'}(v_1), \dots, \pi_{M'}^{N'}(v_n)] \in \text{Dom}(M)$. \square

Example 3.1. *Members of some fitness studio work out on certain weekdays following a particular schedule. This consists of an ordered sequence of exercises on machines. Each exercise is divided into sets of certain numbers of repetitions. Each of these sets is executed with a certain weight. The corresponding nested attribute could be*

Fitness(Member, Week, Day, Schedule[Exercise(Machine, Sets[Rep(Number, Weight)])]).

A small snapshot of a database of this schema is

*(Sylvester,1,Monday,[(Bench,[(10,80),(5,90)],
(Incline,[(8,70),(5,75),(1,80)])]),
(Sylvester,1,Tuesday,[]),
(Sylvester,1,Thursday,[(Bench,[(10,85),(5,85)],
(Incline,[(8,75),(5,75),(1,75)])]),
(Arnold,1,Monday,[(Bench,[(8,100),(5,120)],
(Incline,[(6,70),(4,75),(2,80)])]),
(Arnold,2,Wednesday,[(Bench,[(10,100),(7,120)],
(Incline,[(10,100)])]).* \square

3.3 The Brouwerian Algebra

Fix a set \mathcal{U} of attribute names, and a set \mathcal{L} of labels.

Definition 3.6. Let $N \in \mathcal{NA}$ be a nested attribute. The set $\text{Sub}(N)$ of *subattributes* of N is $\text{Sub}(N) = \{M \mid M \leq N\}$. The *bottom element* λ_N of $\text{Sub}(N)$ is given by $\lambda_N = L(\lambda_{N_1}, \dots, \lambda_{N_k})$ whenever $N = L(N_1, \dots, N_k)$, and $\lambda_N = \lambda$ whenever N is not a tuple-valued attribute. \square

We study the algebraic structure of $\text{Sub}(N)$. A *Brouwerian Algebra* (McKinsey & Tarski 1946) is a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \dashv, 1)$ with top element 1 and a binary operation \dashv which satisfies $a \dashv b \sqsubseteq c$ iff $a \sqsubseteq b \sqcup c$ for all $c \in L$. In this case, the operation \dashv is called the *pseudo-difference*. The *Brouwerian complement* $\neg a$ of $a \in L$ is then defined by $\neg a = 1 \dashv a$. A Brouwerian Algebra is also called a co-Heyting Algebra or a dual Heyting Algebra. The system of all closed subsets of a topological space is a well-known Brouwerian Algebra. It is obvious that $(\text{Sub}(N), \leq, \lambda_N, N)$ is a partially ordered set with bottom element λ_N and top element N .

Definition 3.7. Let $N \in \mathcal{NA}$ and $Y, Z \in \text{Sub}(N)$. The *join* $Y \sqcup_N Z$, *meet* $Y \sqcap_N Z$ and *pseudo difference* $Y \dashv_N Z$ of Y and Z in $\text{Sub}(N)$ are inductively defined as follows:

- $Y \sqcup_N Z = Z$ iff $Y \leq Z$ iff $Y \sqcap_N Z = Y$ and $Z \dashv_N \lambda_N = Z$, and $Z \leq Y$ iff $Z \dashv_N Y = \lambda_N$,
- if $N = L[M]$, $Y = L[A]$, $Z = L[B]$, then $Y \circ_N Z = L[A \circ_M B]$ for $\circ \in \{\sqcup, \sqcap\}$ and if $Z \not\leq Y$, then $Z \dashv_N Y = L[B \dashv_M A]$.
- if $N = L(N_1, \dots, N_n)$, $Y = L(A_1, \dots, A_n)$ and $Z = L(B_1, \dots, B_n)$, then $Y \circ_N Z = L(A_1 \circ_{N_1} B_1, \dots, A_n \circ_{N_n} B_n)$ for $\circ \in \{\sqcup, \sqcap, \dashv\}$. \square

In order to simplify notation, occurrences of λ in a tuple-valued attribute are usually omitted if this does not cause any ambiguities. That is, the subattribute $L(M_1, \dots, M_k) \leq L(N_1, \dots, N_k)$ is abbreviated by $L(M_{i_1}, \dots, M_{i_l})$ where $\{M_{i_1}, \dots, M_{i_l}\} = \{M_j : M_j \neq \lambda_{N_j} \text{ and } 1 \leq j \leq k\}$ and $i_1 < \dots < i_l$. If $M_j = \lambda_{N_j}$ for all $j = 1, \dots, k$, then we use λ instead of $L(M_1, \dots, M_k)$. The subattribute $L_1(A, \lambda, L_2[L_3(\lambda, \lambda)])$ of $L_1(A, B, L_2[L_3(C, D)])$ is abbreviated by $L_1(A, L_2[\lambda])$. However, the subattribute

$L(A, \lambda)$ of $L(A, A)$ cannot be abbreviated by $L(A)$ since this may also refer to $L(\lambda, A)$.

If the context allows, we omit the index N from the operations $\sqcup_N, \sqcap_N, \dashv_N$ and from λ_N . The Brouwerian Algebra of

Schedule[Exercise(Machine, Sets[Rep(Number, Weight)])]

is illustrated in Figure 1. Schedule is abbreviated by J , Exercise by K , Machine by A , Sets by L , Rep by M , Number by B and finally Weight by C .

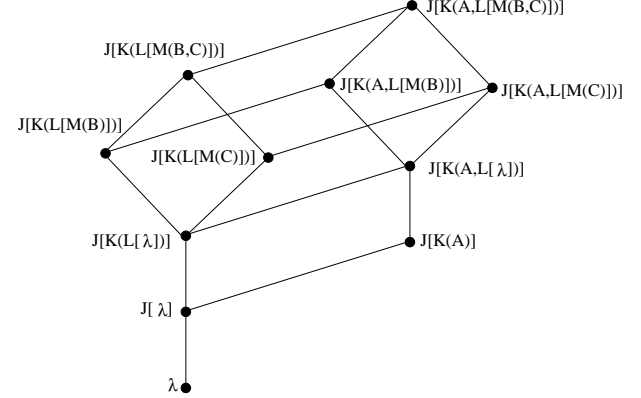


Figure 1: Example of a Brouwerian Algebra

Given some nested attribute $N \in \mathcal{NA}$ and $Y, Z \in \text{Sub}(N)$, we use $Y_N^c = N \dashv Y$ to denote the *Brouwerian complement* of Y in $\text{Sub}(N)$. Again, we omit the subscript N if the context allows. The pseudo difference $Z \dashv Y$ of Z and Y in $\text{Sub}(N)$ satisfies

$$Z \dashv Y \leq X \quad \text{if and only if} \quad Z \leq Y \sqcup X$$

for all $X \in \text{Sub}(N)$. Consequently, for all $X \in \text{Sub}(N)$ holds $Y^c \leq X$ if and only if $X \sqcup Y = N$ holds.

The following result is straightforward to see: $\text{Sub}(\lambda)$ is isomorphic to the Boolean Algebra of order 0, $\text{Sub}(A)$, A a flat attribute, isomorphic to the Boolean Algebra of order 1. $\text{Sub}(L(P))$ is isomorphic to $\text{Sub}(P)$, $\text{Sub}(L(P_1, \dots, P_n))$ isomorphic to the direct product of $\text{Sub}(P_1), \dots, \text{Sub}(P_n)$, and $\text{Sub}(L[P])$ is isomorphic to $\text{Sub}(P)$ augmented by a new minimum. It is an easy exercise to show that the set of all (finite) Brouwerian Algebras is closed with respect to both operations (add a new minimum, direct product). The following theorem generalises the fact that $(\mathcal{P}(R), \subseteq, \cup, \cap, \dashv, \emptyset, R)$ is a Boolean Algebra for a relation schema R in the RDM. This provides us with a powerful framework for the study of various dependency classes in advanced data models.

Theorem 3.1. *(Sub(N), ≤, ⊔_N, ⊓_N, ⋈_N, N) forms a Brouwerian Algebra for every N ∈ NA. \square*

Note that $(\text{Sub}(N), \leq, \sqcup, \sqcap, (\cdot)^c, \lambda, N)$ is in general not boolean. Take for instance $N = L[A]$ and $Y = L[\lambda]$. Then $Y^c = N$ and $Y \sqcap Y^c = Y \neq \lambda$. Furthermore, $Y^{cc} = \lambda \neq Y$. Moreover, every Brouwerian Algebra is distributive.

4 Axiomatising FDs in the Presence of Lists

This section repeats the definition of FDs and their finite axiomatisation proven in (Hartmann & Link & Schewe 2004).

Definition 4.1. Let $N \in \mathcal{NA}$ be a nested attribute. A *functional dependency on N* is an expression of

the form $X \rightarrow Y$ where $X, Y \in \text{Sub}(N)$. A finite set $r \subseteq \text{Dom}(N)$ satisfies a functional dependency $X \rightarrow Y$ on N ($\models_r X \rightarrow Y$) if and only if for all values $t_1, t_2 \in r$ with $\pi_X^N(t_1) = \pi_X^N(t_2)$ always $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ follows. \square

The notions of implication (\models) and derivability ($\vdash_{\mathfrak{R}}$) with respect to a set \mathfrak{R} of inference rules for a class \mathcal{C} of dependencies can be defined analogously to the notions in the RDM (see for instance (Abiteboul & Hull & Vianu 1995, pp. 164-168)). Let Σ be a set of dependencies from \mathcal{C} on some nested attribute N . We are interested in the set of all dependencies in \mathcal{C} implied by Σ , i.e., $\Sigma_{\mathcal{C}}^* = \{\varphi \in \mathcal{C} \mid \Sigma \models \varphi\}$. Our aim is finding sets \mathfrak{R} of inference rules which are sound ($\Sigma_{\mathcal{C}}^+ \subseteq \Sigma_{\mathcal{C}}^*$) and complete ($\Sigma_{\mathcal{C}}^* \subseteq \Sigma_{\mathcal{C}}^+$) for the implication of dependencies in the class \mathcal{C} , and where $\Sigma_{\mathcal{C}}^+ = \{\varphi \in \mathcal{C} \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$. In this paper, we consider the class \mathcal{C} of FDs.

One main result of (Hartmann & Link & Schewe 2004) provides a sound and complete set of inference rules for the implication of FDs in the presence of lists.

Theorem 4.1. *The generalized Armstrong Axioms, i.e.*

$$\frac{}{X \rightarrow Y \leq X}, \quad \frac{X \rightarrow Y}{X \rightarrow X \sqcup Y}, \quad \frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z},$$

called the reflexivity axiom, the augmentation rule and the transitivity rule, form a sound and complete set of inference rules for the implication of FDs in the presence of base, record and finite list types. \square

Proofs of the following lemma and Theorem 4.1 can be found in (Hartmann & Link & Schewe 2004). The lemma will be useful for subsequent studies.

Lemma 4.1. *Let $N \in \mathcal{NA}$. For all $X \in \text{Sub}(N)$ there are $t_1, t_2 \in \text{Dom}(N)$ such that $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ holds if and only if $Y \leq X$.* \square

Example 4.1. *In order to sketch how these rules work we prove the soundness of the join rule*

$$\frac{X \rightarrow Y, X \rightarrow Z}{X \rightarrow Y \sqcup Z}$$

using the following derivation tree

$$\frac{\frac{X \rightarrow Y}{X \rightarrow X \sqcup Y} \quad \frac{\frac{X \sqcup Y \rightarrow X}{X \sqcup Y \rightarrow X} \quad X \rightarrow Z}{X \sqcup Y \rightarrow X \sqcup Y \sqcup Z} \quad X \sqcup Y \rightarrow Y \sqcup Z}{X \rightarrow Y \sqcup Z}.$$

\square

Example 4.2. *We continue Example 3.1 by adding some FDs to the nested attribute. An entry in the database is completely determined by its value on the subattribute $\text{Fitness}(\text{Name}, \text{Week}, \text{Day})$, i.e., members come to the studio at most once a day. Members of our studio follow a strict schedule. They commit to the same sequence of exercises on individual week-days. That is*

$$\text{Fitness}(\text{Name}, \text{Day}) \rightarrow \text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Machine})]).$$

Moreover, the week and sequence of exercises determine the sequence of numbers of sets, i.e., how many sets of the same exercise are performed on each machine.

$$\text{Fitness}(\text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine})]) \rightarrow \text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])]).$$

If we add the member, then this determines even the number of repetitions for each set in the sequence of exercises.

$$\text{Fitness}(\text{Member}, \text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine})]) \rightarrow \text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Number})])]).$$

Note that the snapshot from Example 3.1 satisfies the FDs given. \square

5 Redundancy

It is demonstrated how the notion of redundancy in terms of FDs has to be adapted when the list constructor is present. In order to characterise non-redundant nested attributes a new normal form is proposed which is strictly weaker than a simple extension of BCNF to lists.

5.1 Basis Attributes and Keys

Some further preliminary definitions are given which will be needed later. First we identify the set of those subattributes which is needed to describe any subattribute as the join of some elements of this set.

Definition 5.1. Let $N \in \mathcal{NA}$. The *subattribute basis* $\text{SubB}(N)$ of N is the smallest set $\text{SubB}(N) \subseteq \text{Sub}(N)$ such that for all $X \in \text{Sub}(N)$ we have $X = \sqcup Z$ for some $Z \subseteq \text{SubB}(N)$. Every $X \in \text{SubB}(N)$ is called a *basis attribute* for N . A basis attribute $X \in \text{SubB}(N)$ is called *maximal* if and only if $X \leq Y$ for some basis attribute $Y \in \text{SubB}(N)$ implies that $X = Y$ holds. Basis attributes that are not maximal are called *non-maximal*. Maximal basis attribute of N are denoted by $\text{MaxB}(N)$, non-maximal basis attributes of N by $\text{NMaxB}(N)$. \square

It is immediate that $\lambda \notin \text{SubB}(N)$ since $\lambda = \sqcup \emptyset$. Furthermore, $\text{SubB}(N)$ is not an anti-chain with respect to \leq . Note that any element in $\text{Dom}(N)$ is uniquely determined by its values on all elements in $\text{MaxB}(N)$ (join rule).

Example 5.1. *We determine the subattribute basis of $\text{Schedule}[\text{Exercise}(\text{Machine}, \text{Sets}[\text{Rep}(\text{Number}, \text{Weight})])]$ as*

$$\{ \text{Schedule}[\text{Exercise}(\text{Machine})], \text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Number})])], \text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Weight})])], \text{Schedule}[\lambda], \text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])] \}.$$

Maximal basis attributes are $\text{Schedule}[\text{Exercise}(\text{Machine})]$, $\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Number})])]$ and $\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Weight})])]$. The non-maximal basis attributes are $\text{Schedule}[\lambda]$ and $\text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])]$. The structure of the subattribute basis is illustrated in Figure 2. \square

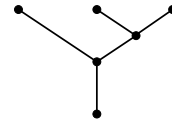


Figure 2: Structure of Subattribute Basis

The concept of keys is very important in relational databases. Given a nested attribute N , a subattribute $X \in \text{Sub}(N)$ is a key whenever the equality of two elements from the domain of N on X already implies the equality of these elements, i.e., values on X uniquely identify an element.

Definition 5.2. Let $N \in \mathcal{NA}$ be a nested attribute and Σ a set of FDs on N . A subattribute $X \in \text{Sub}(N)$ is called a *superkey* for N if and only if $\Sigma \models X \rightarrow N$ holds. In case there is not any proper subattribute X' of X ($X' \leq X$ and $X \neq X'$) which is also a superkey for N , we call X a *minimal key* for N . \square

Obviously, $X \in \text{Sub}(N)$ is a superkey if and only if $X \rightarrow N \in \Sigma^+$ by Theorem 4.1. If $\models_r \Sigma^*$ for some $r \subseteq \text{Dom}(N)$ and X is a superkey for N , then $t_1 = t_2$ whenever $\pi_X^N(t_1) = \pi_X^N(t_2)$ for some $t_1, t_2 \in r$. An FD of the form $X \rightarrow N \in \Sigma^*$ is called a *key dependency* on N if and only if X is a minimal key for N . The set of all key dependencies is denoted by Σ_{key} . Define the closure X^+ of X with respect to Σ as $X^+ = \sqcup \{Z \mid X \rightarrow Z \in \Sigma^+\}$. It follows that X is a superkey for N if and only if $X^+ = N$.

Example 5.2. According to Example 4.2 the subattribute $\text{Fitness}(\text{Member}, \text{Week}, \text{Day})$ is a minimal key for $\text{Fitness}(\text{Member}, \text{Week}, \text{Day}, \text{Schedule}[\text{Exercise}(\text{Machine}, \text{Sets}[\text{Rep}(\text{Number}, \text{Weight})])])$. \square

5.2 Trivial FDs

Suppose we are given a nested attribute N . As in the RDM, there are some FDs on N which are satisfied by any $r \subseteq \text{Dom}(N)$. We call these FDs *trivial*. The *reflexivity axiom* from Theorem 4.1 states that whenever $Y \leq X$ for $X, Y \in \text{Sub}(N)$, then every $r \subseteq \text{Dom}(N)$ satisfies the FD $X \rightarrow Y$. In other words, $X \rightarrow Y$ with $Y \leq X$ is a trivial dependency. One can also show that all trivial FDs have this form. Therefore, assume $X \rightarrow Y$ is such a trivial dependency on N and $Y \not\leq X$ holds. Define $r = \{t_1, t_2\} \subseteq \text{Dom}(N)$ by $\pi_W^N(t_1) = \pi_W^N(t_2)$ if and only if $W \leq X$. Note that this is always possible according to Lemma 4.1. Since $Y \not\leq X$ holds, it follows that $\not\models_r X \rightarrow Y$. This contradicts the triviality of $X \rightarrow Y$. Hence, $Y \leq X$ must hold indeed.

Proposition 5.1. An FD $X \rightarrow Y$ on some nested attribute $N \in \mathcal{NA}$ is trivial if and only if $Y \leq X$ holds. \square

The FD

$$\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Number})])]) \rightarrow \text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])])$$

is a trivial one. Given the sequence of numbers of repetitions for each set and each exercise implicitly tells us the number of sets for each exercise in that sequence.

5.3 Adapting the Notion of Redundancy

In the RDM, the definition of redundancy is based on viewing FDs not only as integrity constraints on a relation, but also as representing the fundamental units of information for retrieving and updating the data in a relation. This interpretation of the semantics of the information stored in a relation was implicit in the original study of normalisation by Codd (Codd 1972), and has since been used in many aspects of database theory. A relation schema is defined to be redundant with respect to a given set of FDs if there exists a relation over the schema which satisfies all these FDs and which has at least two tuples which are identical on a fact. If we formalise this notion of redundancy from the RDM, which goes back to (Beeri & Bernstein & Goodman 1978), in the framework of nested attributes, then we obtain the following definition. Let N be a nested attribute and Σ a set of FDs on N . We call N *redundant with respect to* Σ if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and there are some $t_1, t_2 \in r$ with $t_1 \neq t_2$ and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some non-trivial FD $X \rightarrow Y \in \Sigma$. Intuitively, this notion of redundancy seems to make perfect sense.

Example 5.3. Take a look at the FD

$$\text{Fitness}(\text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine})]) \rightarrow \text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])])$$

from Example 4.2. This is obviously a non-trivial FD. The elements

$$\begin{aligned} &(\text{Sylvester}, 1, \text{Thursday}, [(\text{Bench}, [(10, 85), (5, 85)]), \\ &(\text{Incline}, [(8, 75), (5, 75), (1, 75)])] \text{ and} \\ &(\text{Arnold}, 1, \text{Monday}, [(\text{Bench}, [(8, 100), (5, 120)]), \\ &(\text{Incline}, [(6, 70), (4, 75), (2, 80)])] \end{aligned}$$

coincide on $\text{Fitness}(\text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine}, \text{Sets}[\lambda])])$, i.e., the FD above causes some redundancy according to the definition above. \square

The last example shows that our current definition of redundancy is not really appropriate anymore. That is, the FDs

$$\begin{aligned} &\text{Fitness}(\text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine})]) \rightarrow \\ &\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Number})])]) \text{ and} \\ &\text{Fitness}(\text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine})]) \rightarrow \\ &\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Weight})])]) \end{aligned}$$

are not satisfied by the instance of Example 3.1 and, consequently, redundancy would need to be defined in terms of the non-maximal basis attribute $\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])])$. This, however, appears to be impossible as the information in $\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\lambda])])$ will always be contained in $\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Number})])])$ and $\text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Sets}[\text{Rep}(\text{Weight})])])$. The point here is that the information in a non-maximal basis attribute Y cannot be separated from the information in the maximal basis attribute Z with $Y \leq Z$. This motivates the following definition.

Definition 5.3. Let $N \in \mathcal{NA}$ be a nested attribute and Σ a set of FDs on N . Let $\Sigma_{\text{nasty}} \subseteq \Sigma^+$ denote the set of all $X \rightarrow Y \in \Sigma^+$ where

- $Y \leq X$ holds or
- Y is a non-maximal basis attribute of N .

The elements of the closure Σ_{nasty}^+ of Σ_{nasty} under derivation with respect to the generalized Armstrong Axioms from Theorem 4.1 are called *nasty FDs* on N with respect to Σ . \square

The following lemma characterises nasty FDs which are derivable from a given set of FDs.

Lemma 5.1. Let $N \in \mathcal{NA}$, Σ a set of FDs on N and $X \rightarrow Y \in \Sigma^+$. We have $X \rightarrow Y \in \Sigma_{\text{nasty}}^+$ if and only if every $M \in \text{MaxB}(N)$ with $M \leq Y$ also implies that $M \leq X$.

Proof. Let $X \rightarrow Y \in \Sigma_{\text{nasty}}^+$. Consider the proper chain

$$\Sigma_{\text{nasty}} = \Sigma_0 \subset \Sigma_1 \subset \dots \subset \Sigma_k = \Sigma_{\text{nasty}}^+$$

where Σ_j results from Σ_{j-1} , for $1 \leq j \leq k$, by single application of one of the inference rules from Theorem 4.1. We proceed by induction on j . If $j = 0$ and $X \rightarrow Y \in \Sigma_{\text{nasty}}$, then $Y \leq X$ or $Y \in \text{NMxB}(N)$. Either way, every maximal basis attribute of N which is also a maximal basis attribute of Y must be a maximal basis attribute of X , as well. Assume now that this property holds for all elements in Σ_j for some $j \geq 0$. Consider the single $X \rightarrow Y \in \Sigma_{j+1} - \Sigma_j$. If $X \rightarrow Y$ has been inferred using the *reflexivity axiom*, then $Y \leq X$ and the property holds again. If the *augmentation rule* was used, then $Y = X \sqcup Y'$ and $X \rightarrow Y' \in \Sigma_j$. If $M \in \text{MaxB}(N) \cap \text{MaxB}(Y)$, then $M \in \text{MaxB}(X)$ or $M \in \text{MaxB}(Y')$. In the latter case we can apply hypothesis and conclude that $M \in \text{MaxB}(X)$ as well. It remains to consider the case where $X \rightarrow Y$ has been inferred using the *transitivity rule*, i.e., $X \rightarrow Z, Z \rightarrow Y \in \Sigma_j$. If $M \in \text{MaxB}(N) \cap \text{MaxB}(Y)$, then $M \in \text{MaxB}(Z)$ by hypothesis applied to $Z \rightarrow Y \in \Sigma_j$,

and then $M \in \text{MaxB}(X)$ by hypothesis applied to $X \rightarrow Z \in \Sigma_j$.

We need to show that $X \rightarrow Y \in \Sigma_{\text{nasty}}^+$, if for all $M \in \text{MaxB}(N)$ with $M \leq Y$ also implies that $M \leq X$. Let $Y_1 = \sqcup(\text{MaxB}(Y) \cap \text{MaxB}(N))$. Since $(\text{MaxB}(Y) \cap \text{MaxB}(N)) \subseteq (\text{MaxB}(X) \cap \text{MaxB}(N))$, it follows that $Y_1 \leq X$ and therefore $X \rightarrow Y_1 \in \Sigma_{\text{nasty}}$. Note that $\text{MaxB}(Y) - \text{MaxB}(N) \subseteq \text{NMaxB}(N)$ holds. This implies $X \rightarrow Y' \in \Sigma_{\text{nasty}}$ for every $Y' \in \text{MaxB}(Y) - \text{MaxB}(N)$. This gives $X \rightarrow Y_2 \in \Sigma_{\text{nasty}}^+$ for $Y_2 = \sqcup(\text{MaxB}(Y) - \text{MaxB}(N))$ by the join rule. Applying the join rule again gives $X \rightarrow Y \in \Sigma_{\text{nasty}}^+$ where $Y = Y_1 \sqcup Y_2$ holds. \square

We are now prepared to define a better notion of redundancy for nested attributes *in terms of FDs*.

Definition 5.4. Let N be a nested attribute and Σ a set of FDs on N . We call N *redundant with respect to Σ* if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and there are some $t_1, t_2 \in r$ with $t_1 \neq t_2$ and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some FD $X \rightarrow Y \in \Sigma$ which is not nasty on N with respect to Σ . \square

Example 5.4. With Definition 5.4 of redundancy, the FD

$$\text{Fitness}(\text{Week}, \text{Schedule}[\text{Exercise}(\text{Machine})]) \rightarrow \text{Fitness}(\text{Schedule}[\text{Exercise}(\text{Set}[\lambda])])$$

from Example 4.2 and 5.3 does not cause redundancies anymore, as desired. \square

Note also that we can define two notions of redundancy. Definition 5.4 considers only FDs in Σ itself. As in the RDM, one might define redundancy with respect to all *implied* FDs, i.e., Σ^* . That is, N is called *redundant with respect to Σ^** if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma^*$ and there are some $t_1, t_2 \in r$ with $t_1 \neq t_2$ and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some FD $X \rightarrow Y \in \Sigma^*$ which is not nasty on N with respect to Σ . It can be proven, as in the RDM, that both notions are in fact the same.

Theorem 5.1. Let N be a nested attribute and Σ a set of FDs on N . Then N is redundant with respect to Σ if and only if N is redundant with respect to Σ^* .

Proof. It is easy to see that redundancy of N with respect to Σ is sufficient for the redundancy of N with respect to Σ^+ since $\models_r \Sigma$ implies $\models_r \Sigma^+$ and $\Sigma \subseteq \Sigma^+$. It remains to show that redundancy of N with respect to Σ is also a necessary condition for N to be redundant with respect to Σ^* . Therefore, we assume that N is *non-redundant* with respect to Σ . This means that for all $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and for all $t_1, t_2 \in r$ with $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some $X \rightarrow Y \in \Sigma$, which is not nasty, follows $t_1 = t_2$. We will show that N is non-redundant with respect to Σ^+ . Let therefore

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \dots \subset \Sigma_k = \Sigma^+$$

be a chain where Σ_j results from Σ_{j-1} by application of one of the Armstrong Axioms from Theorem 4.1. What we show, in fact, is that Σ can be replaced by Σ_j . We proceed by induction on j . For $j = 0$ there is nothing to show. Let $j > 0$, i.e., $\Sigma_j - \Sigma_{j-1}$ consists of exactly one dependency $X \rightarrow Y$. If $X \rightarrow Y$ has been inferred using the *reflexivity axiom*, then $Y \leq X$ which means that $X \rightarrow Y$ is trivial and, therefore, also nasty. The non-redundancy of N with respect to Σ_j follows therefore from the hypothesis that N is non-redundant with respect to Σ_{j-1} since there is nothing

to show for $X \rightarrow Y$. Consider the case where $X \rightarrow Y$ has been inferred using the *augmentation rule*, i.e., $Y = X \sqcup Y'$ with $X \rightarrow Y' \in \Sigma_{j-1}$. If $X \rightarrow Y$ is nasty, the statement follows from the hypothesis. Assume therefore that $X \rightarrow Y$ is not nasty and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some $t_1, t_2 \in r$ with $r \subseteq \text{Dom}(N)$ and $\models_r \Sigma_j$. It follows that $\pi_X^N(t_1) = \pi_X^N(t_2)$ and since $\models_r \Sigma_{j-1}$, and in particular $\models_r X \rightarrow Y'$, we obtain that $\pi_{X \sqcup Y'}^N(t_1) = \pi_{X \sqcup Y'}^N(t_2)$ holds. If $X \rightarrow Y'$ was nasty, the join rule would imply that $X \rightarrow Y$ is nasty, too. Therefore, $X \rightarrow Y'$ is not nasty. Now, we can apply the hypothesis and conclude that $t_1 = t_2$ holds. It follows that N is non-redundant with respect to Σ_j . Finally, consider the case where $X \rightarrow Y$ has been derived using the *transitivity rule* with $X \rightarrow Z$, $Z \rightarrow Y \in \Sigma_{j-1}$. Again, we assume that $\models_r \Sigma_j$ for some $r \subseteq \text{Dom}(N)$ and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ holds with $X \rightarrow Y$ not being nasty. Since $\models_r \Sigma_{j-1}$, we conclude that $\pi_{X \sqcup Z}^N(t_1) = \pi_{X \sqcup Z}^N(t_2)$ and $\pi_{Z \sqcup Y}^N(t_1) = \pi_{Z \sqcup Y}^N(t_2)$ hold as well. If $X \rightarrow Z$ and $Z \rightarrow Y$ were both nasty, then $X \rightarrow Y$ would be nasty, too. It follows that at least one of $X \rightarrow Z$ or $Z \rightarrow Y$ is not nasty. In either case we can apply hypothesis, and consequently $t_1 = t_2$. This concludes the proof. \square

5.4 The Nested List Normal Form

The Boyce-Codd Normal Form has been introduced in (Codd 1974) and intensively studied since then. A relation schema R is in BCNF if and only if it is non-redundant with respect to the set of FDs on R . One might therefore say that a well-designed schema should be in BCNF. If N is some nested attribute and Σ a set of FDs on N , then we say that N is in *Boyce-Codd Normal Form (BCNF)* if and only if every $X \rightarrow Y \in \Sigma^*$ is trivial or X is a superkey for N . We might now ask whether BCNF for a nested attribute is a sufficient and necessary condition for the non-redundance of N . Clearly, a nested attribute in BCNF is non-redundant in the sense of Definition 5.4. The converse, however, is false.

Example 5.5. Suppose we have a database for storing the prime factorization of positive integers n .

$$\text{Factor}(\text{Integer}, \text{Prime}[\text{Number}], \text{Exponent}[\text{Number}])$$

would be a suitable nested attribute where *Prime[Number]* is the list of prime factors of n in increasing order, and *Exponent[Number]* the list of exponents for each corresponding prime factor in *Prime[Number]*. A small snapshot of the database could be

$$\begin{aligned} &(12, [2, 3], [2, 1]), \\ &(35, [5, 7], [1, 1]), \\ &(37, [37], [1]), \\ &(936, [2, 3, 13], [3, 2, 1]). \end{aligned}$$

Minimal keys are $\text{Factor}(\text{Integer})$ and $\text{Factor}(\text{Prime}[\text{Number}], \text{Exponent}[\text{Number}])$. Furthermore, we have the FDs

$$\begin{aligned} &\text{Factor}(\text{Prime}[\lambda]) \rightarrow \text{Factor}(\text{Exponent}[\lambda]) \\ &\text{Factor}(\text{Exponent}[\lambda]) \rightarrow \text{Factor}(\text{Prime}[\lambda]). \end{aligned}$$

According to Definition 5.4, the nested attribute is non-redundant with respect to the FDs given. That is, every FD has a superkey on the left-hand side or is nasty. On the other hand, however, N is not in BCNF with respect to the FDs given. \square

Example 5.5 shows that BCNF is not a necessary property for non-redundant nested attributes. Thus, BCNF is too strong to characterise non-redundant nested attributes and, therefore, we would like to find a weaker normal form.

Definition 5.5. Let N be some a nested attribute and Σ a set of FDs on N . We say that N is in *Nested List Normal Form (NLNF)* with respect to Σ if and only if every $X \rightarrow Y \in \Sigma^*$ is a nasty dependency on N with respect to Σ or X is a superkey for N . \square

Note that BCNF implies NLNF, but not vice versa. The nested attribute in Example 5.5 is not in BCNF, but in NLNF. This is a first and important semantic justification for NLNF.

5.5 NLNF - The same fact is only stored once

We show that NLNF captures exactly those nested attributes which are non-redundant in the sense of Definition 5.4.

Theorem 5.2. *Let N be a nested attribute and Σ a set of FDs on N . Then N is non-redundant with respect to Σ if and only if N is in NLNF with respect to Σ .*

Proof. Assume that N is in NLNF. If N was redundant with respect to Σ^* , then there would be some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma^*$ and $t_1, t_2 \in r$, $t_1 \neq t_2$ with $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some FD $X \rightarrow Y \in \Sigma^*$ which is not nasty. In particular, $\pi_X^N(t_1) = \pi_X^N(t_2)$ holds as $X \leq X \sqcup Y$. Since N is in NLNF and $X \rightarrow Y$ is not nasty it follows that X is a superkey for N . This implies $t_1 = t_2$ which is a contradiction. Therefore, N must be non-redundant with respect to Σ^* .

Assume N is non-redundant with respect to Σ^* . Let $X \rightarrow Y \in \Sigma^*$ be an FD which is not nasty. Non-redundance of N with respect to Σ^* implies that $t_1 = t_2$ for all $t_1, t_2 \in r \subseteq \text{Dom}(N)$ with $\models_r \Sigma^*$ and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$. This means that $X \sqcup Y$ is a superkey for N . If $\pi_X^N(t_1) = \pi_X^N(t_2)$ holds for some $t_1, t_2 \in r$, then $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ also holds since $\models_r X \rightarrow Y$. This implies $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ and $t_1 = t_2$ follows. In other words, X is already a superkey for N . Since this is true for every $X \rightarrow Y \in \Sigma^*$ which is not nasty we conclude that N is in NLNF. \square

5.6 Characterising NLNF

Given some nested attribute N and some set Σ of FDs on N , how can we check whether N is in NLNF? That is, how can we check whether N is non-redundant with respect to Σ^* ? By Definition 5.5 one needs to check whether every $X \rightarrow Y$ implied by Σ , i.e., every $X \rightarrow Y$ in Σ^* is a nasty dependency or whether X is a superkey. This is not very practical, although the implication problem for FDs is efficiently decidable, even in the presence of lists. However, we will show now that checking every FD in Σ suffices.

Theorem 5.3. *Let N be a nested attribute and Σ a set of FDs on N . N is in NLNF with respect to Σ if and only if every $X \rightarrow Y \in \Sigma$ is a nasty dependency with respect to Σ or X is a superkey for N .*

Proof. Obviously, if every $X \rightarrow Y$ in Σ^* is a nasty dependency or X is a superkey, then the same is true for every FD in Σ since $\Sigma \subseteq \Sigma^*$. It is therefore sufficient to show that every $X \rightarrow Y$ in Σ^+ has superkey X or is a nasty dependency, if the same is true for every FD in Σ . Consider again the proper chain

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \dots \subset \Sigma_k = \Sigma^+$$

where Σ_j results from Σ_{j-1} , $j > 0$, by a single application of one of the inference rules from Theorem 4.1. We show that there is already some FD in Σ_{j-1} which is not nasty and where X is not a superkey, if there

is some $X' \rightarrow Y'$ in Σ_j which is not nasty and where X' is not a superkey. Let $j > 0$ and $X \rightarrow Y \in \Sigma_j$ not nasty and X not a superkey. Since $X \rightarrow Y$ is not nasty, it is in particular not a trivial dependency. This means $X \rightarrow Y$ has not been derived by the *reflexivity rule* according to Proposition 5.1.

Assume that $X \rightarrow Y$ has been derived by means of the *augmentation rule*, i.e., $Y = X \sqcup Z$ and $X \rightarrow Z \in \Sigma_{j-1}$. Obviously, $X \rightarrow Z$ cannot be nasty since $X \rightarrow Y$ would immediately be nasty too. Moreover, X is not a superkey by assumption. Hence, $X \rightarrow Z \in \Sigma_{j-1}$ is not nasty and X is not a superkey.

Assume that $X \rightarrow Y$ has been derived by means of the *transitivity rule*, i.e., $X \rightarrow Z, Z \rightarrow Y \in \Sigma_{j-1}$. By definition of nasty dependencies, at least one of $X \rightarrow Z, Z \rightarrow Y$ cannot be nasty. On the other hand, neither X nor Z are superkeys. X is not a superkey by assumption. If Z was a superkey, then $X \rightarrow Z, Z \rightarrow N \in \Sigma^+$ and therefore also $X \rightarrow N \in \Sigma^+$ by transitivity. This means that X would be superkey, a contradiction. It is now immediate that one of $X \rightarrow Z, Z \rightarrow Y \in \Sigma_{j-1}$ is neither nasty nor has a superkey on the left-hand side. \square

Theorem 5.3 tells us that NLNF is invariant under derivation (implication) of FDs. This guarantees that one is able to check efficiently whether a given nested attribute is non-redundant with respect to Σ^* . We will now give yet another characterisation of NLNF which will, in particular, give us a different proof of Theorem 5.3. The result extends a well-known result from (Fagin 1981) for relational databases. In order to verify whether a nested attribute N in NLNF satisfies all FDs given, one simply needs to check whether N satisfies all key dependencies and all nasty FDs. This makes integrity checking more efficient and is another justification why nested attributes in NLNF are well-designed. Unlike the RDM where one simply needs to check the satisfaction of all key dependencies for relation schemata in BCNF, one still needs to deal with all nasty FDs when a nested attribute in NLNF is given. This is the price for introducing lists.

Theorem 5.4. *Let N be a nested attribute and Σ a set of FDs on N . N is in NLNF with respect to Σ if and only if every $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$ implies $\models_r \Sigma$.*

Proof. Assume there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$, but $\not\models_r \Sigma$. Then there is some $X \rightarrow Y \in \Sigma$ which cannot be nasty and where X is not a superkey. Since $\Sigma \subseteq \Sigma^*$, N cannot be in NLNF with respect to Σ .

Vice versa, assume that N is not in NLNF with respect to Σ . Then there is some $X \rightarrow Y \in \Sigma^+$ which is not nasty and where X is not a superkey. We show that there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$, but $\not\models_r \Sigma$. We define the closure $X_{\text{nasty}}^+ = \sqcup \{Z \mid X \rightarrow Z \in \Sigma_{\text{nasty}}^+\}$ of X with respect to nasty FDs. According to Lemma 4.1 we define some $r \subseteq \text{Dom}(N)$ with $r = \{t, t'\}$ by

$$\pi_W^N(t') = \pi_W^N(t) \quad \text{if and only if} \quad W \leq X_{\text{nasty}}^+.$$

We show first that $\models_r \Sigma_{\text{key}}$. Let K be an arbitrary minimal key for N . Since X is not a superkey for N we have $X_{\text{nasty}}^+ \leq X^+ \neq N$. This implies that X_{nasty}^+ cannot be a superkey neither. Consequently, $K \not\leq X_{\text{nasty}}^+$, and therefore $\pi_K^N(t') \neq \pi_K^N(t)$ by definition of r .

We show that $\models_r \Sigma_{\text{nasty}}^+$ holds. Let $U \rightarrow V \in \Sigma_{\text{nasty}}^+$. If $U \not\leq X_{\text{nasty}}^+$, then $\pi_U^N(t') \neq \pi_U^N(t)$ and

$\models_r U \rightarrow V$. Suppose $U \leq X_{\text{nasty}}^+$ and, therefore, $\pi_U^N(t') = \pi_U^N(t)$. It follows from the soundness of the join rule that $X \rightarrow X_{\text{nasty}}^+ \in \Sigma_{\text{nasty}}^+$. We have $X_{\text{nasty}}^+ \rightarrow U \in \Sigma_{\text{nasty}}^+$ by reflexivity. Consequently, $X \rightarrow U \in \Sigma_{\text{nasty}}^+$ by transitivity, too. Since $U \rightarrow V \in \Sigma_{\text{nasty}}^+$, we derive $X \rightarrow V \in \Sigma_{\text{nasty}}^+$ as well. This means $V \leq X_{\text{nasty}}^+$ and we conclude $\pi_V^N(t') = \pi_V^N(t)$. Hence, $\models_r U \rightarrow V$.

We show finally that $\not\models_r \Sigma$. If $Y \leq X_{\text{nasty}}^+$ held we would infer $X_{\text{nasty}}^+ \rightarrow Y \in \Sigma_{\text{nasty}}^+$ by reflexivity, and $X \rightarrow Y \in \Sigma_{\text{nasty}}^+$ by transitivity since also $X \rightarrow X_{\text{nasty}}^+ \in \Sigma_{\text{nasty}}^+$ holds. This, however, is a contradiction. Therefore, $Y \not\leq X_{\text{nasty}}^+$ and as $X \leq X_{\text{nasty}}^+$ holds as well, it follows that $\pi_X^N(t') = \pi_X^N(t)$ and $\pi_Y^N(t') \neq \pi_Y^N(t)$. We conclude $\not\models_r \Sigma^*$ and consequently $\not\models_r \Sigma$. \square

According to Theorem 5.4, if N is not in NLNF with respect to Σ , then there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$, but $\not\models_r \Sigma$. This means there is some FD in Σ which is not nasty and where the left-hand side is not a superkey. This gives an alternative proof for Theorem 5.3.

6 Update Anomalies

In the RDM, a relation schema in BCNF does not have any update anomalies. This is another justification why relation schemata should be in BCNF (Bernstein & Goodman 1980). We will demonstrate that nested attributes in NLNF behave very similar. First, the next example reveals a surprising fact.

Example 6.1. Consider Example 5.5 again with *Factor(Integer, Prime[Number], Exponent[Number])*. Recall that this nested attribute is non-redundant with respect to the FD given in Example 5.5. Say our database simply consists of the tuple

$$(12, [2, 3], [2, 1])$$

and the tuple $(35, [5, 7], [1, 1, 0])$ happens to be inserted. Then obviously all key dependencies are still satisfied by the new relation, but the FD

$$\text{Factor(Prime}[\lambda]) \rightarrow \text{Factor(Exponent}[\lambda])$$

is not satisfied. \square

Example 6.1 shows that, in general, the absence of redundancy for a nested attribute does not imply the absence of insertion anomalies. Therefore, it cannot be expected that nested attributes in NLNF do not have update anomalies. We define, however, strong update anomalies in the context of nested attributes. The main difference to the RDM is that updated relations which define any strong anomaly do not only satisfy all key dependencies on the nested attribute, but also all nasty FDs. Deletion anomalies cannot occur with FDs and are therefore not defined.

Definition 6.1. Let N be a nested attribute and Σ a set of FDs on N .

1. We say that N has a *strong insertion anomaly* if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and some $t \notin r$ with $\models_{r \cup \{t\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$, but $\not\models_{r \cup \{t\}} \Sigma$.
2. We say that N has a *strong replacement anomaly*

- of *type 1* if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and some $t \in r$ and $t' \in \text{Dom}(N)$ with $\pi_K^N(t) = \pi_K^N(t')$ for some minimal key K on N and $\models_{r - \{t\} \cup \{t'\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$ and $\not\models_{r - \{t\} \cup \{t'\}} \Sigma$ hold.
- of *type 2* if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and some $t \in r$ and $t' \in \text{Dom}(N)$ with $\pi_K^N(t) = \pi_K^N(t')$ for some distinguished minimal key K on N and $\models_{r - \{t\} \cup \{t'\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$ and $\not\models_{r - \{t\} \cup \{t'\}} \Sigma$ hold.
- of *type 3* if and only if there is some $r \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and some $t \in r$ and $t' \in \text{Dom}(N)$ with $\pi_K^N(t) = \pi_K^N(t')$ for all minimal keys K on N and $\models_{r - \{t\} \cup \{t'\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$ and $\not\models_{r - \{t\} \cup \{t'\}} \Sigma$ hold.

We say that N has a *strong update anomaly* if and only if N has a strong insertion or a strong replacement anomaly of some type. \square

Example 6.2. Consider again the nested attribute

$$\text{Fitness(Member, Week, Day, Schedule[Exercise(Machine, Sets[Rep(Number, Weight)])])$$

and the database from Example 3.1 together with the FDs from Example 4.2. An insertion of the tuple

$$(\text{Sylvester}, 2, \text{Tuesday}, [(\text{Butterfly}, [(20, 50)]))])$$

leads to a new relation which satisfies all key dependencies and all nasty dependencies, but the FD

$$\text{Fitness(Name, Day)} \rightarrow \text{Fitness(Schedule[Exercise(Machine)])}$$

is now violated. Similarly, replacing the tuple

$$(\text{Sylvester}, 1, \text{Thursday}, [(\text{Bench}, [(10, 85), (5, 85)]), (\text{Incline}, [(8, 75), (5, 75), (1, 75)])])$$

by

$$(\text{Sylvester}, 1, \text{Thursday}, [(\text{Bench}, [(12, 85), (6, 85)]), (\text{Incline}, [(10, 75), (5, 75), (1, 75)])])$$

leads to a replacement anomaly of type 1, for instance. The new relation still validates all key dependencies, the new and old tuple coincide on the minimal key $\text{Fitness(Member, Day, Week)}$, but the FD

$$\text{Fitness(Member, Week, Schedule[Exercise(Machine)])} \rightarrow \text{Fitness(Schedule[Exercise(Set[Rep(Number)])])}$$

is violated. \square

The next theorem generalises a result from (Fagin 1981). It shows that NLNF is an exact condition for the absence of strong insertion anomalies.

Theorem 6.1. Let N be a nested attribute and Σ a set of FDs on N . Then is N in NLNF if and only if N does not have any strong insertion anomaly.

Proof. This follows from the proof of Theorem 5.4. In fact, if N has a strong insertion anomaly, then there must be some $X \rightarrow Y \in \Sigma \subseteq \Sigma^+$ which is not nasty and where X is not a superkey. Consequently, N cannot be in NLNF. Vice versa, if N is not in NLNF, then there is some $X \rightarrow Y \in \Sigma^+$ which is not nasty and where X is no superkey. When can now define $t, t' \in \text{Dom}(N)$ exactly as we did in the proof of Theorem 5.4. Take then for instance $r = \{t'\}$ which obviously satisfies $\models_r \Sigma$. The proof of Theorem 5.4 shows then that $\models_{r \cup \{t\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$, but $\not\models_{r \cup \{t\}} \Sigma$. \square

NLNF is also an exact condition for the absence of type 1 replacement anomalies. This is an extension of a well-known result in relational databases (Vincent 1992).

Theorem 6.2. *Let N be a nested attribute and Σ a set of FDs on N . Then is N in NLNF if and only if N does not have any strong replacement anomaly of type 1.*

Proof. Obviously is N not in NLNF if N has a strong replacement anomaly of type 1. Let's assume that N is not in NLNF. Then there is some $X \rightarrow Y \in \Sigma^+$ which is not nasty and where X is not a superkey. It follows by Lemma 5.1 that there is some $M \in \text{MaxB}(N)$ with $M \in \text{MaxB}(Y)$ and $M \notin \text{MaxB}(X)$. That means $X \rightarrow M \in \Sigma^+$ and $X \rightarrow M \notin \Sigma_{\text{nasty}}^+$, again by Lemma 5.1. Let $X_M^+ = \sqcup(\{Z \in \text{SubB}(N) : X \rightarrow Z \in \Sigma^+\} - \{M\})$. Define $t_0, t' \in \text{Dom}(N)$ with

$$\pi_Z^N(t_0) = \pi_Z^N(t') \quad \text{iff} \quad Z \leq X_M^+.$$

Moreover, define $t \in \text{Dom}(N)$ by $\pi_M^N(t) = \pi_M^N(t_0)$ and $\pi_Z^N(t) = \pi_Z^N(t')$ for all $Z \in \text{MaxB}(N) - \{M\}$. Since $M \in \text{MaxB}(N)$ the element t is well-defined. It follows then that

$$\pi_Z^N(t_0) = \pi_Z^N(t) \quad \text{iff} \quad Z \leq X^+.$$

Let $r = \{t_0, t\}$. Show first that $\models_r \Sigma$. Let $U \rightarrow V \in \Sigma$. If $U \leq X^+$, then we need to show that $V \leq X^+$ as well. It follows $X^+ \rightarrow U \in \Sigma^+$ by the reflexivity axiom, $X \rightarrow X^+ \in \Sigma^+$ by the join rule. Applying the transitivity rule a few times shows $X \rightarrow V \in \Sigma^+$. This means, by definition of X^+ , that $V \leq X^+$.

Show next that $\models_{r-\{t\}\cup\{t'\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$. Let K be some minimal key for N . From $K \leq X_M^+$ follows $K \leq X^+$, but X is not a superkey. This is a contradiction, i.e. $K \not\leq X_M^+$ which means that $\pi_K^N(t_0) \neq \pi_K^N(t')$. Let $U \rightarrow V \in \Sigma_{\text{nasty}}^+$. Assume further that $\pi_U^N(t_0) = \pi_U^N(t')$. For all $V' \in \text{MaxB}(V) \cap \text{MaxB}(N)$ follows $V' \in \text{MaxB}(U)$ by Lemma 5.1. This implies that $V' \leq X_M^+$ holds, too. As $\text{MaxB}(V) - \text{MaxB}(N) \subseteq \text{NMaxB}(N)$ holds follows $V' \in \text{NMaxB}(N)$ and therefore $X \rightarrow V' \in \Sigma_{\text{nasty}}$ for all $V' \in \text{MaxB}(V) - \text{MaxB}(N)$. Consequently, $V' \leq X_M^+$ for all $V' \in \text{MaxB}(V) - \text{MaxB}(N)$ ($M \in \text{MaxB}(N)$). Since $V = \sqcup(\text{MaxB}(V) \cap \text{MaxB}(N)) \cup (\text{MaxB}(V) - \text{MaxB}(N))$ we have $V \leq X_M^+$ by means of the join rule.

It is obvious that $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ since $X \leq X_M^+$ ($M \notin \text{MaxB}(X)$), but $M \not\leq X_M^+$, i.e., $\not\models_{r-\{t\}\cup\{t'\}} X \rightarrow M$.

It remains to show that there is some minimal key K such that $\pi_K^N(t) = \pi_K^N(t')$. Let $N_M = \sqcup(\text{MaxB}(N) - \{M\})$. From $X \rightarrow M \in \Sigma^+$ follows $X \sqcup N_M \rightarrow M \sqcup N_M \in \Sigma^+$. This is equivalent to $N_M \rightarrow N \in \Sigma^+$ since $X \leq N_M$ and $M \sqcup N_M = N$. It follows that N_M is a superkey, i.e., there is some minimal key $K \leq N_M$. Since t and t' only differ on M , it follows that $\pi_K^N(t) = \pi_K^N(t')$.

Consequently, there is some $r = \{t_0, t\} \subseteq \text{Dom}(N)$ with $\models_r \Sigma$ and there are $t \in r, t' \in \text{Dom}(N)$ with $\pi_K^N(t) = \pi_K^N(t')$ for some minimal key K for N such that $\models_{r-\{t\}\cup\{t'\}} \Sigma_{\text{key}} \cup \Sigma_{\text{nasty}}^+$ and $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ hold. This means that N has a replacement anomaly of type 1. \square

The following theorem also generalises a well-known result from (Vincent 1992).

Theorem 6.3. *Let N be a nested attribute and Σ a set of FDs on N . Then is N in NLNF if and only if N does not have any strong replacement anomaly of type 2.*

Proof. Obviously is N not in NLNF if N has a strong replacement anomaly of type 2. Let's assume that N is not in NLNF. The existence of some $X \rightarrow M \in \Sigma^+ - \Sigma_{\text{nasty}}^+$ with $M \in \text{MaxB}(N)$, $M \notin \text{MaxB}(X)$ and X is not a superkey for N follows as in the proof of Theorem 6.2. Let K be some distinguished minimal key for N .

If $M \notin \text{MaxB}(K)$, then we can proceed exactly as in the proof of Theorem 6.2. It remains to consider the case where $M \in \text{MaxB}(K)$. Let $Q \leq \sqcup(\text{SubB}(K) - \text{MaxB}(N))$ maximal with respect to \leq and the property that $X \sqcup Q$ is not a superkey. Further define $G = ((X \sqcap K) \sqcup Q)^+$. Define $t_0, t, t' \in \text{Dom}(N)$ with

1. $\pi_Z^N(t) = \pi_Z^N(t_0)$ if and only if $Z \leq G$,
2. $\pi_Z^N(t') = \pi_Z^N(t_0)$ if and only if $Z \leq (X \sqcup G)_{\text{nasty}}^+$,
3. $\pi_K^N(t) = \pi_K^N(t')$.

The first two properties imply that $\pi_G^N(t) = \pi_G^N(t')$, in particular $\pi_{X \sqcap K}^N(t) = \pi_{X \sqcap K}^N(t')$ and $\pi_Q^N(t) = \pi_Q^N(t')$. We show that $\text{SubB}((X \sqcup G)_{\text{nasty}}^+) - \text{SubB}(G)$ is disjoint to $\text{SubB}(K)$. Assume there is some $B \in \text{SubB}((X \sqcup G)_{\text{nasty}}^+) - \text{SubB}(G)$ with $B \in \text{SubB}(K)$. It follows immediately that $B \notin \text{SubB}(X) - \text{SubB}(G)$ since $X \sqcap K \leq G$. This leaves us with $B \in \text{SubB}(K) - \text{SubB}(G)$ and $B \in \text{NMaxB}(N)$, or equivalently $B \in \text{SubB}(K) - \text{MaxB}(N)$ and $B \notin \text{SubB}(G)$. By definition of Q follows that $X \sqcup Q \sqcup B$ is a superkey. From $B \in \text{SubB}((X \sqcup G)_{\text{nasty}}^+)$ follows $B \leq (X \sqcup Q)^+$. Therefore, $X \sqcup Q$ is already a superkey, a contradiction to the choice of Q . The elements t_0, t, t' are well-defined, and in particular can t and t' be chosen to coincide on K .

The claim is that t_0, t, t' define a replacement anomaly of type 2. It is rather easy to show that $\models_{\{t_0, t\}} \Sigma$ and $\models_{\{t_0, t'\}} \Sigma_{\text{nasty}}^+$ hold (the tuples coincide on closed sets, respectively).

Assume t_0 and t' coincide on some minimal key K' . Then $K' \leq (X \sqcup G)_{\text{nasty}}^+$, and $X \sqcup G \rightarrow K' \in \Sigma^+$. This implies that $X \sqcup Q \rightarrow K' \in \Sigma^+$ holds, i.e., $X \sqcup Q$ is some superkey, a contradiction to the choice of Q . Consequently, t_0 and t' differ on every minimal key K' .

It is now sufficient to show that $\not\models_{\{t_0, t'\}} X \rightarrow M$ holds. First of all, $\pi_X^N(t_0) = \pi_X^N(t')$ since $X \leq (X \sqcup G)_{\text{nasty}}^+$. Recall that $M \leq K, M \notin \text{SubB}(X)$, and $M \in \text{MaxB}(N)$. From $M \in \text{MaxB}(N)$ follows $M \not\leq Q$ and therefore $M \not\leq (X \sqcap K) \sqcup Q$. Moreover, if $(X \sqcap K) \sqcup Q \rightarrow M \in \Sigma^+$ held, then K would not be a minimal key. It follows that $M \not\leq G$. From $M \in \text{MaxB}(N), M \notin \text{MaxB}(G), M \notin \text{MaxB}(X)$ and $X \sqcup G \rightarrow M \in \Sigma^+$ follows immediately $X \sqcup G \rightarrow M \notin \Sigma_{\text{nasty}}^+$ by Lemma 5.1. This means $M \not\leq (X \sqcup G)_{\text{nasty}}^+$ and therefore $\pi_M^N(t') \neq \pi_M^N(t_0)$. This concludes the proof. \square

It remains to study strong type 3 replacement anomalies.

Lemma 6.1. *Let N be a nested attribute and Σ a set of FDs on N . If N is in NLNF, then N does not have any strong replacement anomaly of type 3.* \square

Unlike the case for strong type 1 and strong type 2 replacement anomalies, the converse to Lemma 6.1 does not hold in general.

Example 6.3. *Consider the nested attribute*

Paper(Lecturer, Course, Textbook)

together with the FDs

*Paper(Lecturer, Course) \rightarrow Paper(Textbook) and
Paper(Textbook) \rightarrow Paper(Course).*

This nested attribute is not in NLNF with respect to the FDs given. However, the nested attribute does not have any strong replacement anomalies of type 3. In fact, Paper(Lecturer, Course) and Paper(Lecturer, Textbook) are both minimal keys. Consequently, every (modified) tuple t' with $\pi_K^N(t) = \pi_K^N(t')$ for all minimal keys K must be equal to t . This implies immediately that $r - \{t\} \cup \{t'\} = r$ cannot satisfy both $\models_r \Sigma$ and $\not\models_r \Sigma$ simultaneously. \square

The final result follows immediately from the previous theorems.

Theorem 6.4. *A nested attribute in NLNF does not have any strong update anomalies. Strong replacement anomalies of type 1 coincide with strong replacement anomalies of type 2.* \square

The results for strong update anomalies and NLNF are the same as for update anomalies and BCNF.

7 Conclusion

This article has studied normalisation in the presence of lists. The approach is based on a mathematically well-founded algebraic framework based on nested attributes. This framework can easily be extended to other complex object types, thus providing a unifying approach to studying dependencies in various advanced data models. FDs can be captured by a simple generalisation of Armstrong's axioms in this context (Hartmann & Link & Schewe 2004). As in the RDM, FDs cause processing difficulties such as redundancy and abnormal update behavior. The paper has introduced formal definitions for redundancy and various forms of update anomalies in the presence of lists. The main contribution was the proposal of the Nested List Normal Form, a normal form for nested attributes which guarantees the absence of processing difficulties. Indeed, the paper has formally justified the proposal. NLNF is equivalent to the absence of redundancy, insertion anomalies and two types of replacement anomalies. Furthermore, NLNF avoids type 3 replacement anomalies. Moreover, it has been shown that nested attributes can be efficiently tested whether they are in NLNF with respect to a given set of FDs. Finally, integrity checking for nested attributes in NLNF can be reduced to checking all key constraints and all nasty FDs given.

8 Future Work

There are many ways of continuing this research. For the future, we would like to explore richer type systems containing sets, multisets, unions and even references leading to rational trees. The class of FDs in the presence of base, record and finite set types has already been studied in (Hartmann & Link 2003) and

has led to a more sophisticated set of inference rules since the extension rule is no longer valid in the presence of sets. We have also looked at extending this result to lists and multisets.

Another line of research is to extend the classes of dependencies considered. A further important class of dependencies are multi-valued dependencies (MVDs). The work in (Hartmann & Link & Schewe 2004) provides a finite axiomatisation of FDs and MVDs in the presence of lists. This work takes full advantage of the Brouwerian Algebra of subattributes. In (Hartmann & Link 2004) a membership algorithm for deciding the implication of FDs and MVDs has been proposed and proven to work correctly and in polynomial time. In the future, various other classes of relational dependencies (Thalheim 1992), including join dependencies and inclusion dependencies, together with their interactions should be studied with respect to various type systems.

The main objective is of course the study of normal forms for nested attributes which guarantee well-designed databases. The desirable goal would be a theory extending the results from (Vincent 1994) to databases supporting various types. Recall that the Brouwerian Algebra of subattributes can be easily extended to cover sets, multisets, unions etc. The question is then whether the results of this paper can be carried over to different combinations of these types. Another desirable goal is to cover MVDs in an extension of the fourth normal form (4NF) and to semantically justify that extension. The decomposition of nested attributes into NLNF and other normal forms should also be subject to future work. Such a decomposition will, in general, not preserve FDs, and an extension of third normal form and the synthesis approach from the relational data model could be pursued. It might turn out that pivoting (Hartmann 2001) is a more natural approach to decomposition.

References

- Abiteboul, S. & Hull, R. & Vianu, V. (1995): Foundations of Databases, Addison Wesley
- Abiteboul, S. & Buneman, P. & Suci, D. (2000): Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufmann Publishers
- Arenas, M. & Libkin, L. (2002): A Normal Form for XML Documents, *PODS 2002*, ACM
- Armstrong, W.W. (1974): Dependency Structures of database relationships, *Inform. Process.* 74, pp. 580-583
- Batini, C. & Ceri, S. & Navathe, S. B. (1992): Conceptual Database Design: An Entity-Relationship Approach, Benjamin Cummings
- Beeri, C. & Bernstein, P.A. & Goodman, N. (1978): A sophisticated's introduction to database normalization theory. *Proceedings of the 4th International Conference on Very Large Databases*, pp. 113-124
- Beeri, C. & Bernstein, P. A. (1979): Computational Problems Related to the Design of Normal Form Relational Schemas, *ACM Transactions on Database Systems*, Vol. 4, No. 1, pp. 30-59
- Bernstein, P. A. & Goodman, N. (1980): What Does Boyce-Codd Normal Form Do?, *Proceedings of the International Conference on Very Large Databases*, pp. 245-259

- Biskup, J. (1995): Database schema design theory: achievements and challenges, *Proceedings of the 6th International Conference on Information Systems and Management of Data*, Springer, Lecture Notes in Computer Science, Vol. 1066, pp. 14-44
- Biskup, J. (1998): Achievements of relational database schema design theory revisited, *Semantics in databases*, Springer, Lecture Notes in Computer Science, Vol. 1358, pp. 29-54
- Bry, F. & Kröger, P. (2003): A computational biology database digest: data, data analysis, and data management, *Distributed and Parallel Databases*, Vol. 13, Number 1, pp. 7-42
- Chen, P. P. (1976): The Entity-Relationship Model: Towards a unified View of Data, *ACM Transactions Database Systems 1*, 1976, pp. 9-36
- Chen, P. P. (1983): English sentence structure and entity-relationship diagrams, *Information Science 29*, pp. 127-149
- Codd, E. F. (1970): A relational model of data for large shared data banks, *Communications of the ACM*, pp. 377-387
- Codd, E. F. (1972): Further normalization of the database relational model, *Courant Computer Science Symposia 6: Data Base Systems*, Prentice-Hall, pp. 33-64
- Codd, E. F. (1974): Recent investigations in relational database system, *Proceedings of the IFIP Conference*, pp. 1017-1021
- Fagin, R. (1981): A Normal Form for Relational Databases that is based on domains and keys, *Transactions on Database Systems*, Association for Computing Machinery, pp. 387-415
- Gardarin, G. & Cheiney, J.-P. & Kiernan, G. & Pastre, D. & Stora, H. (1989): Managing Complex Objects in an Extensible Relational DBMS, *Proceedings of the Fifteenth International Conference on Very Large Data Bases, August 22-25, Amsterdam, The Netherlands*, Morgan Kaufmann, pp. 55-65
- Hartmann, S. (2001): Decomposing relationship types by pivoting and schema equivalence, *Data and Knowledge Engineering*, Vol. 39, pp.75-99
- Hartmann, S. & Link, S. (2003): More Functional Dependencies for XML, *Lecture Notes in Computer Science*, Volume 2798, pp. 355-369
- Hartmann, S. & Link, S. (2003): On Functional Dependencies in Advanced Data Models, *Electronic Notes in Theoretical Computer Science*, Volume 84
- Hartmann, S. & Link, S. & Schewe, K.-D. (2004): Reasoning about Functional and Multi-valued Dependencies in the Presence of Lists, *Proceedings of the 3rd International Symposium on Foundations of Information and Knowledge Systems, Vienna, Austria*, Lecture Notes in Computer Science
- Hartmann, S. & Link, S. (2004): A Membership Algorithm for Functional and Multi-valued Dependencies in the Presence of Lists, *Electronic Notes in Theoretical Computer*
- Hull, R. & King, R. (1987): Semantic Database Modeling: Survey, Applications and Research Issues, *ACM Computing Surveys*, Vol. 19, Number 3
- McKinsey, J.C.C. & Tarski, A. (1946): On closed elements in closure algebras, *Annals of Mathematics*, Vol. 47, pp. 122-146
- Mok, W.Y. & Ng, Y.K. & Embley, D.W. (1996): A normal form for precisely characterizing redundancy in nested relations, *ACM Transactions on Database Systems*, Vol. 21, 77-106
- Naqvi, S. & Tsur, S. (1989): A logical language for data and knowledge bases, *Computer Science Press*, 1989
- Özsoyoglu, Z.M. & Yuan, L.Y. (1987): A new normal form for nested relations, *ACM Transactions on Database Systems*, Vol. 12, 111-136
- Paredaens, J. & De Bra, P. & Gyssens, M. & Van Gucht, D. (1989): The Structure of the Relational Database Model, *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag
- Richardson, J. (1992): Supporting lists in a data-model, *Proceeding of VLDB*, pp. 127-192
- Schewe, K.-D. & Thalheim, B. (1993): Fundamental Concepts of Object Oriented Databases, *Acta Cybernetica*, Vol. 11, Number 4, pp. 49-85
- Seshadri, P. & Livny, M. & Ramakrishnan, R. (1996): The Design and Implementation of Sequence Database System, *Proceedings of the Twenty-second International Conference on Very Large Data Bases, Mumbai, India*
- Tari, Z. & Stokes, J. & Spaccapietra, S. (1997): Object Normal Forms and Dependency Constraints for Object-Oriented Schemata, *ACM ToDS*, Vol. 22, pp. 513-569
- Thalheim, B. (1991): Dependencies for Relational Databases, *Teubner Verlag*
- Thalheim, B. (1992): Foundations of entity-relationship modeling, *in Ann.Math.Artificial Intelligence 6*, pp. 1-34
- Thalheim, B. (2000): Entity-Relationship Modeling: Foundations of Database Technology, Springer-Verlag
- Tjoa, A. M. & Berger, L. (1993): Transformation of requirement specifications expressed in natural language into an EER model, *Entity-Relationship Approach*, Springer Lecture Notes Series, Vol. 823
- Vincent, M. (1992): Modification anomalies and Boyce-Codd normal form. *In Research and Practical Issues in Databases*. B. Srinivasan and J. Zeleznikow (eds.), World Scientific Press, 251-264.
- Vincent, M. (1994): The semantic justification for normal forms in relational database design, *Monash University*, Melbourne, Australia
- Vincent, M. & Liu, J. & Liu, C. (2003): A redundancy free 4NF for XML, *XML Database Symposium*, 2003
- W3C (2001): XML Schema Part 2: Datatypes, <http://www.w3.org/TR/xmlschema-2/#datatype>