

Domination Normal Form - Decomposing Relational Database Schemas

Henning Koehler

Massey University, Palmerston North, New Zealand
h.koehler@massey.ac.nz

Abstract

A common approach in designing relational databases is to start with a universal relation schema, which is then decomposed into multiple subschemas. A good choice of subschemas can be determined using integrity constraints defined on the schema, such as functional, multivalued or join dependencies.

In this paper we propose and analyze a new normal form based on the idea of minimizing overall storage space. This is in contrast to existing normal forms such as BCNF, 4NF or KCNF, which only characterize the absence of redundancy (and thus space-minimality) for a single schema.

1 Introduction

We begin by introducing some basic terms from relational database theory, followed by a description of the problem we are trying to solve.

1.1 Terminology

A *relation schema* $R = \{A_1, A_2, \dots, A_n\}$ is a set of *attributes*. A *relation* r over a schema R is a set of tuples (where each tuple represents one data item), and each element of the tuple corresponds to one attribute in R .

With each relation schema we associate a set Σ of integrity constraints, in particular *functional dependencies* (FD), *multivalued dependencies* (MVD) and *join dependencies* (JD). These restrict which relations over R we may store. We say that a set Σ of constraints over R *implies* a constraint c , written $\Sigma \models c$, if c holds on every relation r over R for which all constraints in Σ hold.

A FD on R is an expression of the form $X \rightarrow Y$ (read " X determines Y ") where X and Y are subsets of R . We say that a FD $X \rightarrow Y$ *holds* on a relation r over R if every pair of tuples in r that coincides on all attributes in X also coincides on all attributes in Y . We call a FD $X \rightarrow Y$ *trivial* if $Y \subseteq X$, or, equivalently, if it holds on every relation. A set $X \subseteq R$ is a *key* of R if Σ implies $X \rightarrow R$.

For a set $X \subseteq R$ we denote the *projection* of r onto the attributes in X by $r[X]$. The *join* of two relations $r[X]$ and $r[Y]$ is a relation on $X \cup Y$:

$$r[X] \bowtie r[Y] := \left\{ t \mid \begin{array}{l} \exists t_1 \in r[X], t_2 \in r[Y]. \\ t[X] = t_1 \wedge t[Y] = t_2 \end{array} \right\}$$

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at Thirtieth Australasian Computer Science Conference (ACSC2007), Ballarat, Australia. Conferences in Research and Practice in Information Technology, Vol. 62. Gillian Dobbie, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

A join dependency on R is an expression of the form $\bowtie [R_1, \dots, R_n]$ where the R_i are subsets of R with $\bigcup R_i = R$. We say that the JD $\bowtie [R_1, \dots, R_n]$ holds on r if the decomposition $\{R_1, \dots, R_n\}$ is lossless for r , i.e. if

$$r[R_1] \bowtie \dots \bowtie r[R_n] = r$$

A multivalued dependency is a join dependency $\bowtie [R_1, R_2]$ with only two subschemas. It is usually written as $X \twoheadrightarrow Y$ where $X = R_1 \cap R_2$ and $Y = R_1 \setminus R_2$ or (equivalently) $Y = R_2 \setminus R_1$.

1.2 Problems with Existing Normal Forms

When presented with a relation schema R and a set of constraints Σ on it, a designer must decide whether to store all data in one single relation on R , or to decompose R into multiple subschemas. To aid in this task, normal forms are used which characterize "good" solutions.

Many normal forms proposed so far, such as BCNF, 4NF or KCNF, characterize the absence of redundancy. This is desirable for several reasons, foremost the avoidance of update anomalies [4].

However, these normal forms have significant drawbacks. First, they only consider a single relation schema, instead of considering all schemas in a decomposition together.

Definition 1. For a relation r over R and a decomposition $\mathcal{R} = \{R_1, \dots, R_n\}$ of $R = \bigcup R_j$ we denote the decomposition of r by \mathcal{R} as

$$r[\mathcal{R}] := \{r[R_1], \dots, r[R_n]\}$$

where $r[R_j]$ is the projection of r onto the attributes in R_j . When talking about the tuples in $r[\mathcal{R}]$ containing an attribute A , we shall mean the tuples from relations $R_j \in \mathcal{R}$ with $A \in R_j$.

The common generalization to multiple schemas is that the whole schema collection is in that normal form, if every schema taken individually is. But this means that those normal forms cannot capture redundancy which exists across multiple relations. As a trivial example, we can duplicate a schema. Clearly the extra schema is then superfluous in the whole schema collection, but each schema taken individually may still be redundancy free.

Example 1. Let $R = ABCD$ be a schema with constraints $\Sigma = \{AB \rightarrow CD, CD \rightarrow B\}$. Then R is not in BCNF, but has a dependency preserving BCNF decomposition into the subschemas ABC, ABD, BCD .

For any instance r of R , the projections of r onto the schemas ABC and ABD together already take up more space than the original relation r : no tuples are lost in the projection since AB is a key, and

the attributes A and B are stored twice. While we have not defined what redundancy means for multiple schemas, it seems intuitively clear that this decomposition should not be called "redundancy free". From a storage space point-of-view, it is clearly less desirable than the original schema R .

The second big problem is that dependency preserving decompositions into these normal forms do not always exist [2]. Thus, when faced with such a case, a designer must either accept the loss of some dependencies, or cannot use the normal form in question.

We believe that what a normal form should do, is characterize "good" representations (i.e. decompositions) of a schema, in such a way that a "good" representation does always exist. In the following we will propose a normal form which meets this criterion.

2 Minimization as Normal Form

The approach we suggest is the following: among a set of suitable decompositions (e.g. the set of all lossless, or lossless and dependency preserving decompositions), characterize the "best" ones. We do so by defining an order on the decompositions, such that the "best" decompositions are the minimal ones with respect to that order.

This leaves the question of when to call one decomposition better than another one. The motivation for many normal forms proposed so far has been the elimination of redundancy (and with it, the absence of update-anomalies). This may suggest to define a quantitative measure of redundancy over multiple schemas, as has been done in [1].

We take a slightly different route here: instead of trying to minimize redundancy, we try to minimize the size of instances. Intuitively this should lead to similar results, but measures for size appear easier to construct. In the following we will define and motivate three different orders on decompositions, all of which intuitively measure the size.

2.1 Ordering by Size of Instances

Our first approach measures the space required to store an instance. For that we need to know for each element of a domain how much storage space it requires. We represent this knowledge by associating with each domain Dom a size function

$$size : Dom \rightarrow Int$$

Consider e.g. the following domains:

- $STRING$ containing strings of arbitrary length
- $STRING[40]$ containing strings of length up to 40
- INT containing arbitrarily large integers
- $INT(64)$ containing all 64-bit integers
- $BOOLEAN$ containing the values $TRUE$ and $FALSE$

A realistic measure for the size of a string might be its length, the size of an integer i might be defined as $\log(i)$, and the size of $TRUE$ and $FALSE$ might be one.

We shall assume that all domains are infinite, and that the size functions on them are positive and unbounded, i.e. can grow arbitrarily large. This can be justified as follows: For any infinite domain a bounded size function is not realistic, since we can store only a finite number of element when constricted to a fixed

amount of space. While not all domains are truly infinite, they often contain far more elements than the number of subschemas in a typical decomposition (e.g. 256^{40} for $STRING[40]$ or 2^{64} for $INT(64)$). Treating those domains as infinite will allow us to draw a sharp boundary between small (bounded) increases in size from duplicated attributes on one hand, and potentially large (unbounded) increases in size from instances with large numbers of tuples on the other.

We note that this argument fails for domains such as $BOOLEAN$, and that a constant size function may be more realistic for domains such as $STRING[40]$ or $INT(64)$. We won't consider finite domains or constant size functions in this paper though.

As it will turn out, the assumptions about infinite domains and unbounded size functions are all we need to characterize our new normal form, i.e. we do not require detailed knowledge about the actual size functions.

Definition 2. Let $R = \{A_1, \dots, A_k\}$ be a schema. For a finite relation r over R we define the size of r as

$$size(r) := \sum_{t \in r} \sum_{i=1}^k size(\pi_{A_i}(t))$$

We then define the size of the decomposition of r by $\mathcal{R} = \{R_1, \dots, R_n\}$ of R as

$$size(r[\mathcal{R}]) := \sum_{j=1}^n size(r[R_j])$$

While this gives us a suitable definition of size for any instance, we wish to compare schemas w.r.t. the size of all valid instances. If for every valid instance r on R a decomposition \mathcal{R}_1 requires no more storage space than a decomposition \mathcal{R}_2 , then $\mathcal{R}_1 \leq \mathcal{R}_2$ should certainly hold, indicating that \mathcal{R}_1 is "smaller" or "better" than \mathcal{R}_2 . Recall that we shall only consider suitable decompositions, in particular lossless or lossless and dependency preserving ones.

This alone, however, is not sufficient to characterize good decompositions: for an instance r containing only a single element, the trivial decomposition $\{R\}$ requires less storage space than any other lossless decomposition, as those typically need to duplicate some attributes. It would be hard to argue though that decomposition is never necessary. So how can we distinguish decompositions finer, based on the size of instances?

Example 2. Let $R = ABC$ and $\Sigma = \{B \rightarrow C\}$. R can be faithfully decomposed into $\mathcal{R} = \{AB, BC\}$. Clearly every relation r decomposed by \mathcal{R} (which is a set of relations) is at most twice as large as r . On the other hand, for every natural number k we can construct a relation

$$r = \begin{array}{|c|c|c|} \hline A & B & C \\ \hline 1 & 1 & \text{"a very long string"} \\ \hline 2 & 1 & \text{"a very long string"} \\ \hline \vdots & \vdots & \vdots \\ \hline k+1 & 1 & \text{"a very long string"} \\ \hline \end{array}$$

which is more than k times larger than in decomposed form:

$$\begin{array}{|c|c|} \hline A & B \\ \hline 1 & 1 \\ \hline 2 & 1 \\ \hline \vdots & \vdots \\ \hline k+1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline B & C \\ \hline 1 & \text{"a very long string"} \\ \hline \end{array}$$

This observation motivates the following definitions, which compare decompositions similarly to the "big- O " comparison (e.g. $3x^2 + x \in O(x^2)$) from complexity theory.

Definition 3. Let R be a schema with constraints Σ and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . We say that \mathcal{R}_1 *c-dominates* \mathcal{R}_2 (where "c" stands for complexity) if there exists a constant k such that for all finite relations r over R that satisfy Σ we have

$$\text{size}(r[\mathcal{R}_1]) \leq k \cdot \text{size}(r[\mathcal{R}_2])$$

We further say that \mathcal{R}_1 *dominates* \mathcal{R}_2 if the above relation holds for $k = 1$. We abbreviate c-domination and domination as $\mathcal{R}_1 \leq_c \mathcal{R}_2$ and $\mathcal{R}_1 \leq \mathcal{R}_2$, respectively. We say that \mathcal{R}_1 *strictly (c-)dominates* \mathcal{R}_2 , written $\mathcal{R}_1 <_{(c)} \mathcal{R}_2$, if \mathcal{R}_1 (c-)dominates \mathcal{R}_2 but not vice-versa.

It is easy to see that both domination and c-domination are reflexive and transitive, and thus are pre-orders. Clearly domination implies c-domination.

Proposition 1. Let $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . If \mathcal{R}_1 dominates \mathcal{R}_2 then \mathcal{R}_1 c-dominates \mathcal{R}_2 .

Note however that strict domination does *not* imply strict c-domination. Going back to example 2, we see that R and \mathcal{R} are incomparable w.r.t. domination, but \mathcal{R} strictly c-dominates R . In example 1 the original schema $ABCD$ strictly dominates the decomposition $\{ABC, ABD\}$, but both decompositions are equivalent w.r.t. c-domination. Sometimes both criteria, domination and c-domination, are needed to characterize the best decomposition for a schema:

Example 3. Let $R = ABCDE$ be a schema with constraints $\Sigma = \{AB \rightarrow CD, B \rightarrow E\}$. Then the decomposition $\mathcal{R}_1 = \{ABC, ABD, BE\}$ is minimal w.r.t. c-domination but strictly dominated by $\mathcal{R}_2 = \{ABCD, BE\}$. The trivial decomposition $\{R\}$ is minimal w.r.t. domination but strictly c-dominated by both \mathcal{R}_1 and \mathcal{R}_2 .

We are now ready to define our normal form based on the idea of minimizing storage space.

Definition 4. Let R be a schema with constraints Σ and \mathcal{R} be a decomposition of R . We say that \mathcal{R} is in *domination normal form* (DNF) if

- (i) \mathcal{R} is minimal w.r.t. domination, and
- (ii) \mathcal{R} is minimal w.r.t. c-domination

with minimal meaning that no strictly smaller decomposition exists among a given set of 'suitable' decompositions.

Note that this definition depends on the choice which decompositions we consider 'suitable'. We will investigate two different cases (though other choices might be of interest as well): the set of all lossless, and the set of all lossless and dependency preserving decompositions. In each case, we effectively obtain a different DNF.

As the number of suitable decompositions of a given schema is finite, there must exist a decomposition among them which is minimal w.r.t. domination, as well as a (possibly different) schema which is minimal w.r.t. c-domination. It is however not clear yet whether a decomposition into DNF always exists, i.e. one which is minimal w.r.t. both criteria at once. We will show this next.

Theorem 2. *Every schema has a decomposition into DNF.*

Proof. We use the fact that domination implies c-domination. The c-domination pre-order induces a partition of all the (suitable) decompositions of R into equivalence classes and defines a partial order on those equivalence classes. Let EQ be a minimal equivalence class w.r.t. that order. Choose \mathcal{R} to be minimal w.r.t. domination among the decompositions in EQ . We claim that \mathcal{R} is minimal w.r.t. domination among *all* decompositions of R , and thus in DNF. Let \mathcal{R}' be any decomposition with $\mathcal{R}' \leq \mathcal{R}$. Then $\mathcal{R}' \leq_c \mathcal{R}$, and since \mathcal{R} is minimal w.r.t. c-domination, $\mathcal{R}' \in EQ$. But \mathcal{R} is also minimal w.r.t. domination in EQ , and thus $\mathcal{R} \leq \mathcal{R}'$. Thus no decomposition \mathcal{R}' strictly dominates \mathcal{R} . \square

2.2 Ordering by Attribute Count of Instances

Instead of measuring the total size of instances, we could count the number of tuples an attribute appears in. This gives us domination and c-domination pre-orders for each attribute, and we can then combine these to get another pair of orderings for decompositions.

Definition 5. Let $R = \{A_1, \dots, A_k\}$ be a schema. For a finite relation r over R and an attribute A we define the *count* of A on r as

$$\text{count}_A(r) := \begin{cases} |r| & \text{if } A \in R \\ 0 & \text{if } A \notin R \end{cases}$$

where $|r|$ denotes the number of tuples in r . We then define the count of A on r decomposed by a decomposition $\mathcal{R} = \{R_1, \dots, R_n\}$ of R as

$$\text{count}_A(r[\mathcal{R}]) := \sum_{j=1}^n \text{count}_A(r[R_j])$$

Definition 6. Let R be a schema with FDs F , A an attribute and $\mathcal{R}_1, \mathcal{R}_2$ decompositions of R . We say that \mathcal{R}_1 *c-dominates* \mathcal{R}_2 w.r.t. A if there exists a constant k such that for all finite relations r over R that satisfy F we have

$$\text{count}_A(r[\mathcal{R}_1]) \leq k \cdot \text{count}_A(r[\mathcal{R}_2])$$

We further say that \mathcal{R}_1 *dominates* \mathcal{R}_2 w.r.t. A if the above relation holds for $k = 1$.

Thus, for each attribute A , we get a c-domination and domination pre-orders. We shall combine those pre-orders by intersection.

Definition 7. Let R be a schema and $\mathcal{R}_1, \mathcal{R}_2$ decomposition of R . We say that \mathcal{R}_1 $\left\{ \begin{array}{l} \text{dominates} \\ \text{c-dominates} \end{array} \right\}$ \mathcal{R}_2 w.r.t. attribute count if \mathcal{R}_1 $\left\{ \begin{array}{l} \text{dominates} \\ \text{c-dominates} \end{array} \right\}$ \mathcal{R}_2 w.r.t. every attribute.

It will turn out that domination and c-domination w.r.t. attribute count and w.r.t. size are exactly the same orders.

2.3 Ordering by Containing Schema Closures

The previous two order pairs introduced are defined by considering all valid instances. This is not very practical if we wish to actually decide for two given decompositions whether one (c-) dominates the other. We shall therefore give a third pair of orders, which is defined by considering only the decompositions, rather than instances on them.

The approach we use is similar to attribute counting. The count of an attribute depends on the set of schemas it lies in. While it also depends on the instance r , it is easy to show that the number of tuples in $r[R_j]$ is determined by the closure $\overline{R_j}$ of R_j (and r). Recall that the closure $\overline{R_j}$ of R_j under Σ is

$$\overline{R_j} := \{A \in R \mid \Sigma \models R_j \rightarrow A\}$$

where Σ is an arbitrary set of constraints on R (we shall mainly be interested in functional, multi-valued and join dependencies).

Lemma 3. *Let R be a schema with constraints Σ , and $X \subseteq R$. Then for all relations r on R we have $|r[X]| = |r[\overline{X}]|$.*

Proof. We can obtain $r[X]$ by projecting from $r[\overline{X}]$. We could only lose tuples if $r[\overline{X}]$ contains different tuples which are identical on X . But this can't happen since X functionally determines \overline{X} . \square

This motivates the following definitions.

Definition 8. Let R be a schema with constraints Σ , and \mathcal{R} a decomposition of R . Then for any attribute $A \in R$ we define the *containing schema closures* (CSC) of A in \mathcal{R} as the *multiset*

$$CSC_A(\mathcal{R}) = \{\overline{R_j} \mid A \in R_j \in \mathcal{R}\}$$

Note that it is necessary to use multisets rather than sets to obtain the correct attribute count.

Example 4. Consider again the schema $R = ABCDE$ with constraints $\Sigma = \{AB \rightarrow CD, B \rightarrow E\}$ from example 3, and the decompositions

$$\mathcal{R}_1 = \{ABC, ABD, BE\}$$

$$\mathcal{R}_2 = \{ABCD, BE\}$$

They produce the multisets

$$CSC_A(\mathcal{R}_1) = \{ABCDE, ABCDE\}$$

$$CSC_A(\mathcal{R}_2) = \{ABCDE\}$$

which indicate that, for any relation r on R , the attribute A appears in twice as many tuples in $r[\mathcal{R}_1]$ as in $r[\mathcal{R}_2]$. Using sets would hide this difference.

We shall now compare decompositions using the respective CSCs of all attributes. For that we need mappings between multi-sets. We shall allow different instances of the same value in the source domain to map to different values, and call a mapping injective if a value in the target domain is mapped to at most as often as it occurs in the target domain.

Definition 9. Let M_1, M_2 be two multisets¹ of (attribute) sets. We say that

(i) M_1 *weakly inclusion-dominates* M_2 if there exists a mapping $f : M_1 \rightarrow M_2$ with $e \subseteq f(e)$ for all $e \in M_1$.

(ii) M_1 *strongly inclusion-dominates* M_2 if there exists an injective mapping $f : M_1 \rightarrow M_2$ with $e \subseteq f(e)$ for all $e \in M_1$.

Definition 10. Let $\mathcal{R}_1, \mathcal{R}_2$ be two decompositions of R . We say that \mathcal{R}_1 *weakly/strongly csc-dominates* \mathcal{R}_2 if for all attributes $A \in R$ we have that $CSC_A(\mathcal{R}_1)$ weakly/strongly inclusion-dominates $CSC_A(\mathcal{R}_2)$.

It will turn out that weak csc-domination implies c-domination, and that strong csc-domination implies domination. While the opposite does not hold for arbitrary types of constraints, we will be able to show that it holds for sets of functional dependencies, and in the case of weak csc-domination/c-domination also for multi-valued and join dependencies.

¹Note that for definition (i) we do not really need multi-sets.

3 Equivalence of Orderings

We will show that, if the only constraints on the schema are functional, multi-valued and join dependencies, then all three order pairs defined in section 2 are identical (with one possible exception which we will discuss later). As multi-valued dependencies are just a special case of join dependencies, it suffices to consider only functional and join dependencies. We will assume throughout this section that functional and join dependencies are the only types of integrity constraints occurring.

3.1 Size vs. Attribute Count

We start by showing that the orders defined by size and attribute count are identical. Recall that we assume that all domains are infinite, and that the size functions associated with them are positive and unbounded.

Lemma 4. *Let R be a schema with constraints Σ , and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . If \mathcal{R}_1 does not (*c*-)dominate \mathcal{R}_2 w.r.t. attribute count, then \mathcal{R}_1 does not (*c*-)dominate \mathcal{R}_2 w.r.t. size.*

Proof. If \mathcal{R}_1 does not *c*-dominate \mathcal{R}_2 w.r.t. attribute count, then for every integer k there exists a relation r and an attribute A with

$$count_A(r[\mathcal{R}_1]) > k \cdot count_A(r[\mathcal{R}_2])$$

For non-domination such r and A only need to exist for $k = 1$. For each k and associated r and A , we shall construct a relation r' for which

$$size(r'[\mathcal{R}_1]) > k \cdot size(r'[\mathcal{R}_2])$$

holds. This shows non-(*c*-)domination w.r.t. size, thus proving the lemma.

The construction works as follows. Since the relations in $r[\mathcal{R}_1]$ with attribute A contain more than k times as many tuples as those in $r[\mathcal{R}_2]$, there must be an attribute value v_A for A which appears more than k times as often in $r[\mathcal{R}_1]$ as in $r[\mathcal{R}_2]$.

We construct r' from r by substituting every occurrence of v_A by a new value v'_A which doesn't appear in r . As Σ contains only functional and join dependencies, these constraints still hold for r' . Let o_1, o_2 be the number of occurrences of v_A in $r[\mathcal{R}_1], r[\mathcal{R}_2]$. We choose v'_A sufficiently large, i.e. such that

$$size(v'_A) > \frac{k \cdot size(r[\mathcal{R}_2]) - size(r[\mathcal{R}_1])}{o_1 - k \cdot o_2} + size(v_A)$$

This gives us (note that $o_1 - k \cdot o_2 > 0$):

$$(o_1 - k \cdot o_2) \cdot (size(v'_A) - size(v_A)) > k \cdot size(r[\mathcal{R}_2]) - size(r[\mathcal{R}_1])$$

$$size(r[\mathcal{R}_1]) + o_1 \cdot (size(v'_A) - size(v_A)) > k \cdot size(r[\mathcal{R}_2]) + k \cdot o_2 \cdot (size(v'_A) - size(v_A))$$

$$size(r'[\mathcal{R}_1]) > k \cdot size(r'[\mathcal{R}_2])$$

\square

Lemma 5. *Let R be a schema with constraints Σ , and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . If \mathcal{R}_1 does not (*c*-)dominate \mathcal{R}_2 w.r.t. size, then \mathcal{R}_1 does not (*c*-)dominate \mathcal{R}_2 w.r.t. attribute count.*

Proof. The proof is analogous to the last one. For every k, r ($k = 1$ for domination) with

$$size(r[\mathcal{R}_1]) > k \cdot size(r[\mathcal{R}_2])$$

we need to construct a relation r' such that for some attribute A we get

$$count_A(r'[\mathcal{R}_1]) > k \cdot count_A(r'[\mathcal{R}_2]).$$

Again we use that the attribute values occurring in $r[\mathcal{R}_1]$ and $r[\mathcal{R}_2]$ are the same. Since $size(r[\mathcal{R}_1]) > k \cdot size(r[\mathcal{R}_2])$, there must exist some attribute value v_A of an attribute A which occurs more than k times as often in $r[\mathcal{R}_1]$ than in $r[\mathcal{R}_2]$. We construct r' from r by selecting exactly those tuples which have the value v_A on attribute A . As Σ contains only functional and join dependencies, these constraints still hold for r' . And clearly we now have $count_A(r'[\mathcal{R}_1]) > k \cdot count_A(r'[\mathcal{R}_2])$. \square

We can combine the last two lemmas.

Theorem 6. *Let R be a schema with constraints Σ , and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . Then \mathcal{R}_1 (c -)dominates \mathcal{R}_2 w.r.t. size iff \mathcal{R}_1 (c -)dominates \mathcal{R}_2 w.r.t. attribute count.*

Proof. Follows immediately from the lemmas shown. \square

3.2 Attribute Count vs. Containing Schema Closures

We shall now show that the orders defined by attribute count and containing schema closures are actually the same. One direction of implication is easy to show.

Theorem 7. *Let R be a schema with constraints Σ , and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R .*

If \mathcal{R}_1 $\left\{ \begin{array}{l} \text{weakly} \\ \text{strongly} \end{array} \right\}$ csc-dominates \mathcal{R}_2 then \mathcal{R}_1 $\left\{ \begin{array}{l} c\text{-dominates} \\ \text{dominates} \end{array} \right\}$ \mathcal{R}_2 .

Proof. Let r be any relation on R and A some attribute in R .

If \mathcal{R}_1 weakly csc-dominates \mathcal{R}_2 , then for every schema $R_1 \in \mathcal{R}_1$ with $A \in R_1$ there exists a schema $R_2 \in \mathcal{R}_2$ with $A \in R_2$ and $\overline{R_1} \subseteq \overline{R_2}$. By lemma 3 we have $|R_1| \leq |R_2|$, and each such schema R_2 is mapped to at most $|R_1|$ times. Therefore the number of tuples containing attribute A in $r[\mathcal{R}_1]$ is at most $|R_1|$ times larger than the number of tuples with A in $r[\mathcal{R}_2]$. Thus \mathcal{R}_1 c -dominates \mathcal{R}_2 with $k = |R_1|$.

If \mathcal{R}_1 strongly csc-dominates \mathcal{R}_2 , then by lemma 3 and due to the injectivity of the mapping f in definition 9, the number of tuples with attribute A in $r[\mathcal{R}_1]$ is no larger than the number of those in $r[\mathcal{R}_2]$. Thus \mathcal{R}_1 dominates \mathcal{R}_2 . \square

It is possible to show implication in the other direction. This can be done by assuming that \mathcal{R}_1 does not weakly or strongly csc-dominate \mathcal{R}_2 , and constructing example relations which show that \mathcal{R}_1 does not c -dominate or dominate \mathcal{R}_2 . These constructions are extensive though, and we will omit them here. Instead we only present the results.

Theorem 8. *Let R be a schema with functional and join dependencies Σ , and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . If \mathcal{R}_1 does not weakly csc-dominate \mathcal{R}_2 , then \mathcal{R}_1 does not c -dominate \mathcal{R}_2 .*

Theorem 9. *Let R be a schema with functional dependencies Σ , and $\mathcal{R}_1, \mathcal{R}_2$ be decompositions of R . If \mathcal{R}_1 does not strongly csc-dominate \mathcal{R}_2 , then \mathcal{R}_1 does not dominate \mathcal{R}_2 .*

4 Relationship to other Normal Forms

A number of normal forms have been proposed for relational databases, depending on the types of integrity constraints given. For functional dependencies, BCNF and 3NF are the most popular ones. For functional and multivalued dependencies 4NF is the logical extension of BCNF, which for functional and join dependencies has then been extended to PJ/NF, 5NFR and KCNF. The last normal form, KCNF, will be of particular interest to us.

Definition 11. Let R be a schema with functional and join dependencies Σ . Then R is in *Key-Complete Normal Form (KCNF)*, if for every join dependency $\bowtie [R_1, \dots, R_n]$ implied by Σ , the keys among R_1, \dots, R_n cover R . That is, we have

$$\bigcup \{R_i \mid \Sigma \models R_i \rightarrow R\} = R$$

Note that in the case where Σ contains only functional and multivalued dependencies, KCNF is equivalent to 4NF, and when Σ contains only functional dependencies KCNF equates to BCNF [6].

Given a single schema with constraints Σ , the absence of redundancy is precisely characterized by BCNF, 4NF and KCNF. That is, a schema R is free of redundancy iff it is in BCNF, 4NF or KCNF [6].

While our normal form has been designed to minimize size rather than redundancy, the intuition is that minimizing one minimizes the other as well. We will show that this intuition holds insofar, as that when we consider the set of all lossless decompositions, then a single schema is in DNF iff it is in KCNF. Thus DNF can be seen as an extension of BCNF, 4NF and KCNF (when considering all lossless decompositions).

Recall that a decomposition is in DNF if it is minimal among a given set of 'suitable' decompositions, and that for each such set we obtain a different DNF.

Theorem 10. *Let R be a schema with constraints Σ . Then R is in KCNF iff $\{R\}$ is in DNF w.r.t. all lossless decompositions of R .*

Proof. (1) Let R be in KCNF. Let $\mathcal{R} = \{R_1, \dots, R_n\}$ be any lossless decomposition of R . Then Σ implies the join dependency $\bowtie [R_1, \dots, R_n]$, and since R is in KCNF, every attribute $A \in R$ lies in some R_i which forms a key of R . Thus $R \in CSC_A(\mathcal{R})$ for all A , so $\{R\}$ strongly csc-dominates \mathcal{R} . Therefore $\{R\}$ dominates \mathcal{R} by theorem 7. As this holds for all lossless decompositions \mathcal{R} , $\{R\}$ is in DNF.

(2) Let R not be in KCNF. Then Σ implies a join dependency $\bowtie [R_1, \dots, R_n]$ such that

$$\bigcup \{R_i \mid \Sigma \models R_i \rightarrow R\} \neq R$$

The decomposition $\mathcal{R} = \{R_1, \dots, R_n\}$ is lossless, and there exists an attribute $A \in R$ which does not lie in any R_i which forms a key of R . Clearly \mathcal{R} weakly csc-dominates $\{R\}$, and since $R \notin CSC_A(\mathcal{R})$ we have that $\{R\}$ does not weakly csc-dominate \mathcal{R} . Thus \mathcal{R} strictly c -dominates $\{R\}$ by theorems 7 and 8, showing that $\{R\}$ is not in DNF. \square

5 An Example

A university has oral examinations at the end of each semester, and wants to manage related data using a relational database. The relevant attributes to be stored are

$$R = \{Student, Course, Chapter, Time, Room\}$$

Here Chapter denotes a chapter from the course textbook the student will be examined about. Every student can get examined about multiple chapters, and chapters may vary for each student. Multiple students can get examined at the same time in the same room, but the course must be the same. Further constraints are that a student gets examined for a course only once, and can't be in multiple rooms at the same time. Those conditions can be expressed through functional dependencies as follows:

$$\Sigma = \left\{ \begin{array}{l} \{Student, Course\} \rightarrow Time, \\ \{Student, Time\} \rightarrow Room, \\ \{Time, Room\} \rightarrow Course \end{array} \right\}$$

We are now presented with the task of decomposing R . If it is deemed necessary to preserve dependencies, a reasonable Boyce-Codd Normal Form decomposition could be synthesized as follows:

$$\mathcal{R}_{DP-BCNF} = \left\{ \begin{array}{l} \{Student, Course, Time\}, \\ \{Student, Time, Room\}, \\ \{Course, Time, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

This decomposition, however, is not in dependency preserving Domination Normal Form - it is strictly dominated by the decomposition

$$\mathcal{R}_{DP-DNF} = \left\{ \begin{array}{l} \{Student, Course, Time, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

Note that the latter decomposition is in dependency preserving DNF (although we won't show this here), but not in BCNF.

If dependencies need not be preserved, we could use the well-known BCNF decomposition algorithm [3, 4, 5] to obtain the following BCNF decomposition (decomposing first by $\{Student, Course\} \rightarrow Time$, then by $\{Student, Course\} \rightarrow Room$):

$$\mathcal{R}_{BCNF} = \left\{ \begin{array}{l} \{Student, Course, Time\}, \\ \{Student, Course, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

Again, this is strictly dominated by the decomposition \mathcal{R}_{DP-DNF} , which is in DNF even among all lossless decompositions. At first look it might seem that \mathcal{R}_{DP-DNF} is strictly c-dominated by

$$\mathcal{R}_{DNF} = \left\{ \begin{array}{l} \{Student, Time, Room\}, \\ \{Course, Time, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

A closer look reveals though that both c-dominate each other, since the attribute $Course$ already appears in the key schema $\{Student, Course, Chapter\}$ in both cases. The latter decomposition is both in DNF and BCNF. The decomposition

$$\mathcal{R}'_{DP-DNF} = \left\{ \begin{array}{l} \{Student, Course, Time, Room\}, \\ \{Student, Time, Chapter\} \end{array} \right\}$$

however, while in dependency preserving DNF, is not in DNF w.r.t. all lossless decomposition, since it is strictly c-dominated by

$$\mathcal{R}'_{DNF} = \left\{ \begin{array}{l} \{Student, Time, Room\}, \\ \{Course, Time, Room\}, \\ \{Student, Time, Chapter\} \end{array} \right\}$$

Which decomposition to choose is ultimately up to the designer, as storage space and redundancy are

not the only design criteria to consider. The schema $\{Student, Course, Chapter\}$ appears to be a more intuitive choice than $\{Student, Time, Chapter\}$, although deciding this requires domain knowledge which is not encoded in the constraints.

We conclude this example by providing an instance of R and its projection onto the first four decompositions given. While DNF considers all valid instances rather than just a given one, we chose an instance which visualizes the relationship between a schema's closure and the size of relations projected onto it.

<i>Student</i>	<i>Course</i>	<i>Ch.</i>	<i>Time</i>	<i>Ro.</i>
J.C. Denton	Networks	2	3/10, 1pm	101
J.C. Denton	Networks	6	3/10, 1pm	101
J.C. Denton	Security	1	4/10, 1pm	104
J.C. Denton	Security	5	4/10, 1pm	104
L. Nasher	Networks	3	3/10, 1pm	101
L. Nasher	Networks	4	3/10, 1pm	101
L. Nasher	Security	4	4/10, 1pm	104
L. Nasher	Security	7	4/10, 1pm	104
O. Shrek	Networks	2	3/10, 1pm	101
O. Shrek	Networks	8	3/10, 1pm	101
O. Shrek	Security	5	4/10, 1pm	104
O. Shrek	Security	2	4/10, 1pm	104
M. Smith	Security	4	4/10, 2pm	104
M. Smith	Security	6	4/10, 2pm	104
M. Anderson	Networks	3	3/10, 1pm	101
M. Anderson	Networks	5	3/10, 1pm	101
A. Cheng	Networks	2	3/10, 1pm	103
A. Cheng	Networks	4	3/10, 1pm	103
A. Cheng	Security	4	4/10, 2pm	104
A. Cheng	Security	5	4/10, 2pm	104
N. Cheng	Networks	1	3/10, 1pm	103
N. Cheng	Networks	7	3/10, 1pm	103
N. Cheng	Security	5	4/10, 2pm	104
N. Cheng	Security	6	4/10, 2pm	104
J.Zhao	Networks	2	3/10, 1pm	103
J.Zhao	Networks	5	3/10, 1pm	103

All four decompositions share the key schema $\{Student, Course, Chapter\}$, with corresponding projection

<i>Student</i>	<i>Course</i>	<i>Ch.</i>
J.C. Denton	Networks	2
J.C. Denton	Networks	6
J.C. Denton	Security	1
J.C. Denton	Security	5
L. Nasher	Networks	3
L. Nasher	Networks	4
L. Nasher	Security	4
L. Nasher	Security	7
O. Shrek	Networks	2
O. Shrek	Networks	8
O. Shrek	Security	5
O. Shrek	Security	2
M. Smith	Security	4
M. Smith	Security	6
M. Anderson	Networks	3
M. Anderson	Networks	5
A. Cheng	Networks	2
A. Cheng	Networks	4
A. Cheng	Security	4
A. Cheng	Security	5
N. Cheng	Networks	1
N. Cheng	Networks	7
N. Cheng	Security	5
N. Cheng	Security	6
J.Zhao	Networks	2
J.Zhao	Networks	5

The decompositions differ only by the remaining schemas. We obtain the projections:

<i>Student</i>	<i>Course</i>	<i>Time</i>	<i>Ro.</i>
J.C. Denton	Networks	3/10, 1pm	101
J.C. Denton	Security	4/10, 1pm	104
L. Nasher	Networks	3/10, 1pm	101
L. Nasher	Security	4/10, 1pm	104
O. Shrek	Networks	3/10, 1pm	101
O. Shrek	Security	4/10, 1pm	104
M. Smith	Security	4/10, 2pm	104
M. Anderson	Networks	3/10, 1pm	101
A. Cheng	Networks	3/10, 1pm	103
A. Cheng	Security	4/10, 2pm	104
N. Cheng	Networks	3/10, 1pm	103
N. Cheng	Security	4/10, 2pm	104
J.Zhao	Networks	3/10, 1pm	103

<i>Student</i>	<i>Course</i>	<i>Time</i>
J.C. Denton	Networks	3/10, 1pm
J.C. Denton	Security	4/10, 1pm
L. Nasher	Networks	3/10, 1pm
L. Nasher	Security	4/10, 1pm
O. Shrek	Networks	3/10, 1pm
O. Shrek	Security	4/10, 1pm
M. Smith	Security	4/10, 2pm
M. Anderson	Networks	3/10, 1pm
A. Cheng	Networks	3/10, 1pm
A. Cheng	Security	4/10, 2pm
N. Cheng	Networks	3/10, 1pm
N. Cheng	Security	4/10, 2pm
J.Zhao	Networks	3/10, 1pm

<i>Student</i>	<i>Time</i>	<i>Ro.</i>
J.C. Denton	3/10, 1pm	101
J.C. Denton	4/10, 1pm	104
L. Nasher	3/10, 1pm	101
L. Nasher	4/10, 1pm	104
O. Shrek	3/10, 1pm	101
O. Shrek	4/10, 1pm	104
M. Smith	4/10, 2pm	104
M. Anderson	3/10, 1pm	101
A. Cheng	3/10, 1pm	103
A. Cheng	4/10, 2pm	104
N. Cheng	3/10, 1pm	103
N. Cheng	4/10, 2pm	104
J.Zhao	3/10, 1pm	103

<i>Student</i>	<i>Course</i>	<i>Ro.</i>
J.C. Denton	Networks	101
J.C. Denton	Security	104
L. Nasher	Networks	101
L. Nasher	Security	104
O. Shrek	Networks	101
O. Shrek	Security	104
M. Smith	Security	104
M. Anderson	Networks	101
A. Cheng	Networks	103
A. Cheng	Security	104
N. Cheng	Networks	103
N. Cheng	Security	104
J.Zhao	Networks	103

<i>Course</i>	<i>Time</i>	<i>Ro.</i>
Networks	3/10, 1pm	101
Security	4/10, 1pm	104
Security	4/10, 2pm	104
Networks	3/10, 1pm	103

Clearly \mathcal{R}_{DP-DNF} and \mathcal{R}_{DNF} requires less storage space than $\mathcal{R}_{DP-BCNF}$ and \mathcal{R}_{BCNF} .

6 Conclusion

We have introduced a new normal form called DNF for relational databases, based on the type of constraints given and the requirements for a decomposition. Here, a decomposition is in DNF if and only if there is no "better" decomposition among the decompositions considered. The partial orders describing this "better" property have been defined in semantical terms, and for functional and in part for join dependencies, they have been characterized syntactically. Using this syntactical characterization we then showed that, when considering all lossless decompositions, our normal form is an extension to the existing normal forms BCNF, 4NF and KCNF, which have already proven to be useful over the past 30 years.

For multiple schemas DNF appears more suitable than just considering schemas individually, as has been done traditionally. At the same time it is always applicable, in that a decomposition into DNF always exists, even when we restrict ourselves to certain types of decompositions, e.g. dependency preserving ones.

References

- [1] ARENAS, M., AND LIBKIN, L. An information-theoretic approach to normal forms for relational and xml data. In *PODS (2003)*, pp. 15–26.
- [2] BEERI, C., AND BERNSTEIN, P. A. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems* 4, 1 (1979), 30–59.
- [3] LEVENE, M., AND LOIZOU, G. *A Guided Tour of Relational Databases and Beyond*. Springer, 1999.
- [4] MAIER, D. *The Theory of Relational Databases*. Computer Science Press, 1983.
- [5] MANNILA, H., AND RÄIHÄ, K.-J. *The Design of Relational Databases*. Addison-Wesley, 1987.
- [6] VINCENT, M. W. Redundancy elimination and a new normal form for relational database design. In *Semantics in Databases (1998)*, vol. 1358 of *Lecture Notes in Computer Science*, Springer, pp. 247–264.