Atomicity and Normalization

Andy Carver and Terry Halpin

Neumont University, Utah, USA.
e-mail: {andy, terry}@neumont.edu

Abstract: A common aim of data modeling approaches is to produce schemas whose instantiations are always redundancy-free. This is especially useful when the implementation target is a relational database. This paper contrasts two very different approaches to attain a redundancy-free relational schema. The Object-Role Modeling (ORM) approach emphasizes capturing semantics first in terms of atomic (elementary or existential) fact types, followed by synthesis of fact types into relation schemes. Normalization by decomposition instead focuses on "nonloss decomposition" to various, and progressively more refined, "normal forms". Nonloss decomposition of a relation requires decomposition into smaller relations that, upon natural join, yield the exact original population. Nonloss decomposition of a table *scheme* (or relation variable) requires that the decomposition of all possible populations of the relation scheme is reversible in this way. In this paper we show that the dependency requirement for "all possible populations" is too restrictive for definitions of multivalued and join dependencies over relation schemes. By exploiting modeling heuristics underlying ORM, we offer better definitions of these data dependencies, and of "nonloss decomposition", thus enabling these concepts to be addressed at a truly semantic level.

1 Introduction

In relational database design, being able to achieve a fully normalized schema is generally considered desirable, mainly because relations are then guaranteed to be free of redundancy, thus simplifying the process of maintaining consistency as the database is updated. The acceptance of that value-premise is in fact the starting point of the current paper. The question which this paper addresses is not, whether we need a procedure for producing normalized relation schemes, but rather, which procedure is both effective and most appropriate for achieving this desired result[1].

The question does not have an obvious answer: indeed, various approaches are recommended. Conceptual data modeling approaches such as Entity-Relationship Modeling (ER) and Object-Role Modeling (ORM) use a two phase process: *conceptualization*, in which information is first portrayed in terms of conceptual schemas suitable for communication with domain experts [22], and then *deconceptualization* where these structures are mapped into relational schemas.

[1] While some situations may require denormalization for performance reasons, these are best handled by starting with a normalized schema and then adapting it as needed, applying constraints to ensure controlled redundancy [e.g.,13, pp. 642-647].

In contrast, the *normalization* approach to database design ignores conceptualization, instead representing information directly in terms of relational database structures, such as relation schemes (i.e. relation variables) and various dependencies. This paper's treatment of normalization focuses on normalization by *decomposition*, ignoring normalization by synthesis[2]. Normalization by decomposition basically follows a process of achieving progressively higher levels of normalization (called "*normal forms*") through "nonloss decomposition" of given relational table schemes. Just how the original tables became "given" in the first place, the procedure does not say.

The ER approach captures data in terms of entity attributes and relationships between entities, and then applies a mapping procedure to transform these structures into a relational database schema [e.g., 2]. The ORM approach captures information in terms of atomic fact types, and then applies an algorithm such as Rmap to map these fact types and associated constraints into a relational schema [13, 18]. ORM is a prime example of the *fact-oriented* modeling approach, which uses the fact type (relationship type) as its sole data structure. Features modeled as attributes in ER (e.g., Person.birthdate) are modeled in ORM as relationships (e.g., Person was born on Date). Other examples of fact-orientation include Natural language Information Analysis Method (NIAM) [23] and the Predicator Set Model (PSM) [21]. Overviews of ORM may be found in [14, 15], and a detailed coverage in [13].

Nonloss decomposition of a relation requires decomposition into smaller relations that, upon natural join, yield the exact original population. Nonloss decomposition of a table scheme requires that the decomposition of all possible populations of the relation scheme is reversible in this way. In this paper we show that the dependency requirement for "all possible populations" is too restrictive for definitions of multivalued and join dependencies over relation schemes. Unlike ORM's conceptual-schema-design-and-relational-mapping procedure, the traditional normalization procedure neither seeks, nor invokes the concept of, "atomic" fact types, and this is the source of its problem. By exploiting the fact-oriented nature and modeling heuristics of ORM, we offer better, more accurate definitions of these data dependencies, and of "nonloss decomposition", thus enabling these concepts to be addressed at a truly semantic level.

Section 2 reviews the traditional notions of nonloss decomposition and data dependency in normalization theory. Section 3 illustrates the failure of the accepted definitions of multivalued dependency and 4th normal form. Section 4 solves these problems by defining a semantic notion of nonloss decomposition, and applies this notion to define semantic versions of multivalued and join dependencies that overcome the defects in the commonly accepted notions of 4th and 5th normal form. Section 5 summarizes the main contributions and lists references.

[2] Normalization by synthesis assumes as input the set of all attributes and functional dependencies, and provides an algorithm to group the attributes into relation schemes. As such input is typically unavailable in practice, the synthesis technique is mainly of academic interest and has been largely ignored by practitioners. Its main transforms, however, have analogous mapping transforms in ORM (e.g., all FDs are determined by semantic uniqueness constraints).

2 The Traditional Version of the Normalization Procedure

As mentioned above, the traditional version of the normalization procedure comprises a process of achieving progressively higher levels of normalization (called "normal forms") through "nonloss decomposition" of given relational table schemes. In mathematics, the term "relation" means a set of ordered *n*-tuples. In relational databases [3], tuples are ordered by name (by pairing value entries with their attribute name). By the Principle of Extensionality, sets are determined by their membership, and hence are fixed (unchanging). So in relational databases a *relation* is basically a table *population* (fixed set of tuples). In contrast, a *relation scheme* is a table structure or table *variable*, whose population may vary at different times. A relation scheme is sometimes referred to as a relation schema [6] or "relvar" [5]. We now explain what is meant by "nonloss decomposition".

2.1 The traditional definition of "nonloss decomposition"

The notion of "decomposition" involved here means, the breaking up of a table scheme, through relational projection, into smaller schemes, the union of whose headings includes all the attributes of the original scheme, and whose headings are usually overlapping, so that natural joins might be performed on the smaller, resulting schemes. The notion of "nonloss" involved here means: any facts recorded in (a tuple in) the original table may still be retrieved, and therefore no information (or in other words, no question-answering ability) has been lost by doing the decomposition.

This is not to say that there is any worry that tuples will be lost from the original table: on the contrary, all the original tuples will definitely be retrieved in (what we might call) any "recomposition" through natural join. The concern is rather that some scheme-decompositions result in *spurious* (i.e. *extra, non-original) tuples* occurring in the relation that results from the natural join; and this inability to reproduce exactly the original table-population is said to compromise our question-answering ability, or as it's usually said, "information has been lost". After all, the population resulting from the natural join might just as well have been the result of decomposition-and-recomposition performed on a different starting population. Thus, the traditional definition of "nonloss" (or "lossless") decomposition could be stated as, a decomposition of the table-scheme such that it is guaranteed that, *for any population of the scheme*, the decomposition will be reversible, that is, a natural join on the relations resulting from the decomposition would produce the same, exact, original population as was decomposed [5, p. 353; 17, pp. 374ff.].

What we need then, it is alleged, is a way to predict what table-scheme decompositions will be "nonloss" according to the above definition. And this question gets its traditional answer from the theory of data dependencies.

2.2 Data dependencies (as traditionally defined)

Normalization theory defines various population-tuple-patterns, called "data dependencies", which guarantee the feasibility of this sort of "nonloss decomposition". The original dependency, which Codd defined in 1971 [4], was "functional dependency" (FD). Later, Fagin defined "multivalued dependency" (MVD) [7], and at about the same time, others defined the more general "join dependency" (JD) [1]. If we can determine that every possible population of a given relation scheme has one of these data dependencies, and that it is a "nontrivial" dependency and not enforced by a key-constraint on the relation scheme, then (according to the traditional normalization theory) we can perform on this relation scheme a "nonloss decomposition", and we should do so, in order to approach the desired goal of a normalized relational schema.

As an example of a decomposition based on a nontrivial MVD, consider Fig. 1, where the intended semantics is "Person (identified by Surname) plays Sport (identified by SportName) and speaks Language (identified by LanguageName)".

Surname	Sport	Language
Halpin	*judo*	*English*
Halpin	*karatedo*	*English*
Halpin	*judo*	*Japanese*
Halpin	*karatedo*	*Japanese*
Carver	*judo*	*English*

Fig. 1. A relation with a nontrivial MVD between Surname and Sport/Language

If we decompose this relation (i.e., population) by doing a relational projection on {Surname, Sport} and one on {Surname, Language}, we get the relations in Fig. 2. Performing a natural join on these relations reproduces exactly the relation of Fig. 1.

Surname	Sport
Halpin	*judo*
Halpin	*karatedo*
Carver	*judo*

Surname	Language
Halpin	*English*
Halpin	*Japanese*
Carver	*English*

Fig. 2. Relations from decomposing Fig. 1's on {Surname, Sport} and {Surname, Language}

While that relation is thus nonloss-decomposable into two smaller relations, some relations cannot be so decomposed into two relations, but can be so decomposed into some larger number of relations (e.g., 3) [1]. This more general case of decomposability is called a (nontrivial) "join dependency". An MVD is a special case of a JD, and an FD is a special case of an MVD.

2.3 Distinguishing scheme-dependency and population-dependency

As Date has emphasized [5, p. 65], "it is an unfortunate fact that much of the literature uses the term *relation* when what it really means is a relation *variable* (as well as when it means a relation *per se*—i.e., a relation *value*). Historically, however, this practice has certainly led to some confusion.... most of the current database literature still fails in this respect". Indeed, it may be said that even Codd [4, pp. 34-35, 62-63] and Fagin [8, p. 266; 9, p. 534] seem rather guilty of this; Fagin assumed (rather than argued) that all that was needed, in order to make the population-based definitions of "FD" and "MVD" relevant to table schemes as found in a real database, was to (specify, as usual, the relation's heading, and then) claim the relation may be "time-varying" (i.e. a time-varying set of tuples). Codd also defined schemes in this way.

Date is more careful in his thinking here, in defining a table scheme as really a relation-*variable* (a.k.a. *relvar*), of a particular relation-type. And this added level of indirection opens up scope for more attention to the relation's heading, and in particular to the meaning of that heading, which is in fact a predicate [5, pp. 65-67, 129]. By contrast, Codd's and Fagin's treatments of dependencies were mathematically pristine yet included only cursory consideration, if any, of the table scheme's semantics.

One importance of this relation-vs.-relvar distinction for the current discussion is that it clarifies our question: We want to know, not the meaning and importance of some generic thing called a "data dependency", but rather, we want to know what characteristic(s) of a table *scheme* – or, if (but only if) it be found relevant, of "every possible *population"* of a table scheme – allows us to pronounce it normalized, or not. Also, it clarifies that what traditional normalization theory as pioneered by Codd, Fagin, et al, has defined as "data dependencies", are *tuple*-patterns, in *relations* (i.e. table *populations*), not in *relvars* (i.e. table *schemes*). *This leaves it an open question*, whether the connection by which scheme-dependencies have been defined in terms of population-dependencies is really valid (viz., that in order for the *scheme* to have the dependency in question, *every possible population* of the scheme must have the corresponding *relation*-dependency: "corresponding" here means, the dependencies at both levels must be of the same class: FD, MVD, or JD). But as just suggested, what we ultimately need is to be able to detect which *schemes* may be nonloss decomposed.

3 Failure of the Traditional Normalization Procedure

And that, as it turns out, is where this version of the normalization procedure fails. The failure of this approach to defining "scheme-dependency" can be very easily illustrated for MVDs. An analogous but more general illustration would apply to JDs.

3.1 Illustrating the failure of the definitions of 'MVD' and (thus) '4NF'

Let us reconsider the relation of Fig. 1. Suppose that for whatever reason, we delete from this relation the particular tuple listed fourth in Fig. 1, leaving the population shown in Fig. 3.

Surname	Sport	Language
Halpin	*judo*	*English*
Halpin	*karatedo*	*English*
Halpin	*judo*	*Japanese*
Carver	*judo*	*English*

Fig. 3. A smaller version (i.e. fewer tuples) of the relation of Fig. 1

Given our earlier interpretation of this scheme, "Person plays Sport and speaks Language", there is still fact-redundancy in the population, even after removal of this tuple (e.g., the fact that Halpin plays judo is stored twice). However, once this tuple has been removed, the remaining population is lacking that particular tuple-data-pattern which is the criterion for presence of a (nontrivial) MVD. There being no nontrivial MVDs in this population, and if we accept this as one "possible population" of this relation *scheme* (which pragmatically it clearly is), then it follows that there is a possible population of this table scheme lacking any nontrivial MVD, and thus, the table scheme is in Fourth Normal Form (4NF) despite this fact-redundancy.

This perhaps surprising result follows from Fagin's "constructive" characterization of multivalued dependencies, which stipulated that if certain tuples are present in a relation that satisfies an MVD, then certain other tuples must appear also. According to Fagin [8], given a relation $R(X, Y, Z)$ where X, Y, and Z are attribute sets, the MVD $X \twoheadrightarrow Y$ holds if and only if, whenever (x, y_1, z_1) and (x, y_2, z_2) are tuples of R then so are (x, y_2, z_1) and (x, y_1, z_2). For further discussion, see Fagin & Vardi [10].

Since the whole purpose of defining MVDs and 4NF was to avoid fact-redundancy, the traditional definitions of scheme-MVD and (thus) of 4NF have failed to identify correctly the phenomenon causing the problem. If someone suggested the table in Fig. 3 for the design of a database, we would want some way of checking whether the design was sensible, but normalization theory fails to help. That also brings into question the validity of the traditional criterion of "nonloss decomposition", since when interpreted conjunctively (as here) the relation scheme populated in Fig. 3 clearly *can* be semantically decomposed into two smaller relation schemes.

3.2 What went wrong? A research-historical, psychological excursus

When such a fairly obvious error in a standard, accepted theory goes undetected for three decades, one cannot help but ask what went wrong. It is worth noting that Codd's original paper on normalization [4] neither defined any concept of "nonloss decomposition", nor stated any criterion of it; it simply gave an example of such a decomposition, and said, "No essential information has been lost, since at any time the original relation T may be recovered by taking the natural join ... of T_1 and T_2...". It nowhere stated that this was a necessary, and not merely a sufficient, condition of information-nonloss. Claiming this as a *criterion* of nonloss decomposition thus seems to have been the contribution of others.

The problem is, it could be a necessary condition of information-nonloss only if no one had ever discovered any other sorts of dependency (past FD). As soon as population-MVDs or population-JDs were discovered, the definition of "nonloss decomposi-

tion" became in need of emendation. Thus, even more interesting than the question what went wrong originally, is the question why no one noticed it for so long.

It is interesting to survey the random variety of ways in which database texts try to handle – or, more often, simply overlook or ignore – this surprising error, in their discussions of MVDs and 4NF. For the only logical way to rebut the above critique of the standard definition of scheme-MVD, would be to say that the population of Fig. 3 is *not* a legal, "possible population" of this table scheme – even though it obviously is. In fact, this is exactly what Date [5, p. 353] and Elmasri and Navathe [7, p. 437] try to do, but with defective, and differing, artificial arguments, discussion of which we relegate to a footnote.[3] Other writers generally either mention, yet forego any discussion of, MVD and 4NF (e.g., [2], [17],); or they provide as an example of nontrivial MVD a relation like that in Fig. 3, claiming (contrary to the definition of 4NF they had just given) that it is *not* in 4NF, and thus needs decomposing (e.g., [20]); or else they offer such a relation, with its obvious fact-redundancy, yet argue that since it has no nontrivial MVDs it presents no problem (e.g., [19])! Thus researchers have used varying, but ultimately failed, ways to treat the topic of MVDs.

Such varied approaches suggest that there is no single logical error that most researchers fell into with regard to population-MVDs and 4NF. More likely, the true explanation is that by the time MVDs and JDs were discovered (in 1977), everyone simply "knew" what nonloss decomposition entailed; and the idea that their assumption about this was just flat wrong, was too radical a thought to occur to anyone.

---

[3] Date [5] does avoid using a 4NF example table and calling it non-4NF; his example relation does have a nontrivial MVD, as Fagin defines the latter. However, Date gives a poor excuse for restricting himself to such an example: "You might suggest that [the relvar] CTX need not include all possible teacher/text combinations for a given course; for example, two tuples are obviously sufficient to show that the physics course has two teachers and two texts. The problem is, *which* two tuples? Any particular choice leads to a relvar having a very unobvious interpretation and very strange update behavior (try stating the predicate for such a relvar! – i.e., try stating the criteria for deciding whether or not some given update is an acceptable operation on that relvar)" (pp. 391-92). That is weak: the predicate is statable easily, as "Course uses Text and has Teacher"; and so if the text *is* used by that course and the course *has* that teacher, then why, logically, shouldn't the update be accepted?

Elmasri and Navathe [7], like Date, offer an unrealistic justification for limiting themselves to a truly non-4NF example relation (p. 437; the italics are ours):

> In Figure 13.4(a) the MVDs ENAME→→PNAME and ENAME→→DNAME, or ENAME→→PNAME/DNAME hold in the EMP relation. The employee with ENAME 'Smith' works on projects with PNAME 'X' and 'Y' and has two dependents with DNAME 'John' and 'Anna'. *If we stored only the first two tuples in EMP (<'Smith', 'X', 'John'> and <'Smith', 'Y', 'Anna'>), we would incorrectly show associations between project 'X' and 'John' and between project 'Y' and 'Anna';* these should not be conveyed, because no such meaning is intended in this relation. Hence, we must store the other two tuples (<'Smith', 'X', 'Anna'> and <'Smith', 'Y', 'John'>) *to show that {'X', 'Y'} and {'John', 'Anna'} are associated only with 'Smith'; that is, there is no association between PNAME and DNAME.*

This is spurious reasoning. In the first place, storing the other two tuples does not at all *rule out* the possibility of an "association between PNAME and DNAME". And conversely, not including them does not *imply* such an association.

A contributing factor to this oversight, however, seems to have been the aforementioned, and mathematicians' natural, tendency to focus on the syntax – to the neglect of semantics, in this case. The definition of "nonloss decomposition" solely in terms of population-tuple-patterns and recovery of the exact, original *population*, even though their purported goal was to avoid loss of *information* (a sort of *meaning*), says something about the syntax-focused nature of most early researchers' mindset. And as we have seen, Date and Fagin conducted essentially all their investigations (pre-1978, anyway) at the population level, neglecting the distinct, scheme-level.

And yet, there is a significant difference between these two, respective levels: The relation scheme level has, as an *essential* part of its makeup, an expression of the *meaning* of the predicate that its relation-heading represents. And it is easily demonstrable that the basic question, "Does this relation scheme suffer from uncontrolled fact-redundancy", pivots fundamentally on the *semantics* of the relation scheme, and not on the syntax of its populations. Let us reconsider the relation of Fig. 3, but this time ascribe to it the alternate semantics, "*Person plays Sport only if it is refereed in Language*". By assigning it those different semantics, we eliminate the fact-redundancy which *the same relation, with the same heading,* had previously. Thus, *depending only on the semantics,* the relation scheme qualifies as normalized or else as non-normalized (assuming we take "normalized" to mean: lacking any potential fact-redundancy not fully controllable by table-scheme key-constraints).

4 Redefining Nonloss Decomposition and Scheme-MVD, -JD

We must, however, return to the traditional definition of "nonloss decomposition". For Date indeed had an argument for that definition—although he considered his argument so noncontroversial that he relegated it to a footnote! [5, p. 353]: he claimed that if the recomposition (after the decomposition) does not reproduce the exact, original pre-decomposition population, "we have no way in general of knowing which tuples [in the natural-join table] are spurious and which genuine, [therefore] we have indeed lost information". The question arises: is this argument sound?

Let us test it on the example we used in Fig. 3. If we decompose this relation on the attribute-sets {Surname, Sport} and {Surname, Language}, we get in fact the two tables shown in Fig. 2. Doing a natural join of these tables, we get back, not the original table (of Fig. 3), but the table displayed in Fig. 1, repeated for convenience in Fig. 4 with its additional (non-original) tuple shown in boldface.

Surname	Sport	Language
Halpin	*judo*	*English*
Halpin	*karatedo*	*English*
Halpin	*judo*	*Japanese*
Halpin	***karatedo***	***Japanese***
Carver	*judo*	*English*

Fig. 4. The relation resulting from natural join of the tables in Fig. 2

To facilitate further discussion, we introduce the notion of relation *transparency*. With respect to a given state of the universe of discourse (UoD), we say that a relation is *transparent* if and only if each candidate tuple (of that relation) that is composed of attribute values present in the relation and encodes a fact true of that UoD state, is explicitly present in the relation. This is similar to our notion of semiclosed fact types [16, ch. 10], except that the extension is relative to the role populations not the object type populations. For example, with respect to the relation scheme in Fig. 4, a relation composed of no rows, or just the first row, or just the first two rows, is transparent. Assuming the relation scheme means "Person plays Sport and speaks Language", the relation composed of just the first three rows is not transparent, since it does not include the fourth row (which is known to be true given the underlying semantics and the second and third rows).

Now, is Date correct, that we have (now) no way of knowing which of the tuples populating this relation scheme are true and which are spurious? Before answering this, let's specify the intended semantics, since this seems potentially relevant. First, let's ask the question based on the assumption of our newer semantics, "Person plays Sport only if it is refereed in Language". Even if we knew which rows were original (and not simply, that the original rows were true), we could not tell whether this additional tuple was true, without knowing whether, and indeed that, the original population was transparent.

But if we ascribe to the relation scheme our earlier semantics for it, "Person plays Sport and speaks Language", it is clear that the new, additional tuple must be true, if the original ones were. But what is it that is importantly different, about the meaning ascribed? It is that this meaning may be split into two predicate meanings without loss of information, because the original predicate's meaning is that of a logical conjunction, and the meanings of the predicates into which we split it were those of its (two) conjuncts. Examination of the tuples shows us that if we know the conjunctions expressed by the original tuples were true, then we know that the conjunction expressed in the additional one is true, because we can see from the four original tuples that both *conjuncts* are true which, conjoined, make up the conjunction asserted by the fifth, added tuple.

This suggests a better definition for "nonloss decomposition". For it is manifest that *if and only if* we can determine that the original relation scheme *is conjunctive in meaning*, we can know that it is nonloss-decomposable. (Finding a non-key-based, nontrivial FD in a relation scheme is one way of implying that its meaning is conjunctive.) Nor do we assume too much, by assuming that we can tell whether its meaning is conjunctive: Codd's and Date's discussions clearly assume that we have access to the meaning of the relations, e.g., we know which populations are "possible" for the domain, and which tuples assert truths.

Having found the true criterion for "nonloss decomposition", may we continue to use the traditional definitions of scheme-MVD, 4NF, etc.? No, we cannot continue to use the definition of scheme-MVD, since it requires "every possible population" of the relation scheme to have the (population-)MVD, which we have now seen is *not* a necessary (though it be a sufficient) condition for MVD-based decomposability. Nor can we continue to use the traditional definition of 4NF, since it is defined in terms of that traditional, incorrect sense of "scheme-MVD".

Is this a problem, though? For we have now found a completely effective way to tell whether a table-scheme is nonloss-decomposable, and one which has nothing to do with finding population-MVDs or -JDs. So do we even *need* a definition for scheme-MVD? Do we really even *need* to think about 4NF?

So as not to beg our question of which approach to ensuring normalization is most efficient, let us emend so as to correct, if possible, traditional definitions of scheme-MVD and scheme-JD. We may generalize this correction to the definition of scheme-FD as well:

> *scheme { functional | multivalued | join } dependency:*
>
> There is a scheme {functional | multivalued | join} dependency over a sequence of attribute sets in relation scheme *R** if and only if for every possible state of the UoD, each **transparent** relation *R* that instantiates *R** has a relational dependency of the same kind over those arguments.

This definition-pattern does indeed give us conditions both sufficient and necessary for a table-scheme's being nonloss-decomposable due to its having a pattern involving the corresponding relation-level data-dependency. However, if this emendation is of theoretic significance, it seems nevertheless of very little practical significance; for the crucial addition of transparency to the definition-pattern makes the definition *unusable* (except for FDs, as we shall see) *apart from a prior, independent knowledge that the meaning of the relation scheme is conjunctive!*

So, practically speaking, database design might as well forget about MVDs and 4NF—as well as JDs and 5NF—and focus instead on the question: is this relation scheme's predicate-meaning conjunctive? However, as practitioners and theoreticians may still want to say things about particular patterns of non-normalization that can give rise to fact-redundancy, one may adopt the above definition-pattern for scheme data dependencies, and give these new concepts that follow this definition-pattern some special name, such as "*semantic MVD*", "*semantic 4NF*", and so on.

However, someone might object that this redefinition does too much, since the definition of scheme-FD we have used for all these years worked just fine—i.e., the definition which did *not* include that segment boldfaced in the above definition-pattern. If we need this boldfaced interpolation in the definition, why didn't we have to have it in there before, to define FD in a valid way? The answer is twofold: First, even though defining FD without this interpolated clause gives a correct definition, it is not really as good a way to define FD, since it doesn't really define by essentials: the essential question pertains, not to every possible population, but only to those particular populations, for each respective, possible state of the universe of discourse, that are transparent with respect to that relation scheme.

Second, an FD is a special case of MVD, one whose functional nature makes it a constraint on each individual tuple in the relation, and not just on the relation (population) as a whole; thus, mere removal of a tuple cannot possibly eliminate a population-FD, whereas it very easily, as we have seen, may eliminate a population-MVD or -JD. Thus, if an FD applies to each population that is transparent for a given relation scheme, it applies also to any other population that is true of this same state of the UoD. That is the only reason the traditional definition of FD actually worked. But be-

cause it worked, it also gives us a way to detect conjunctive (and hence nonatomic) predicates in other, less direct ways, e.g. comparing the arity of the candidate keys with the arity of the relation scheme, or looking for scheme-FDs that are not implied by any whole-candidate-key constraint (i.e., "embedded" FDs).

A rational procedure for ensuring fully normalized relation schemes should thus begin by decomposing the relation schemes into atomic (and hence nonconjunctive) relation schemes, by looking for conjunction in the given predicates (a necessarily informal process that can be reliably performed only by a domain expert), and double-checking this by looking for nontrivial FDs of certain forms incompatible with atomicity, as is done in ORM's Conceptual Schema Design Procedure (CSDP) [13].

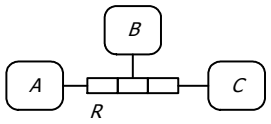
In ORM, a *fact* is a proposition taken to be true by the community of users in the business domain. An *atomic fact* is either an elementary fact or an existential fact. An *elementary fact* is essentially the instantiation of a typed logical predicate that is irreducible in the sense that it cannot be (re)phrased as a conjunction of simpler facts with the same object types. For example, Person was born in Country is an elementary fact type. An *existential fact* is an assertion that an object exists (e.g., There is a Country that has CountryCode 'AU').

From an ORM perspective, each tuple of a relation encodes one or more atomic facts. In step 1 of ORM's CSDP, the domain expert verbalizes information examples (e.g. table rows in an output report) in natural language sentences, and the modeler rephrases the information in terms of atomic facts, checking with the domain expert that the meaning is as intended. Part of this step includes the conjunction check: Can this sentence be (re)phrased as a conjunction of simpler sentences with the same object terms? While the presence of conjunctive connectors (e.g., "and") can help in answering this question, such a presence is neither necessary nor sufficient, since natural language is not formally regulated. Another requirement for atomicity is the absence of nulls in sample fact populations (if a null occurs in a fact tuple, the remaining non-null portion corresponds to a smaller fact, so the original fact is nonatomic).

Knowledge of constraints sometimes helps. For example, the ternary fact type Flight goes from Airport to Airport can be seen to be nonatomic because of FDs from the flight role to each airport role. ORM detects such cases using checking procedures such as its *n*-1 rule which provides a sufficient (though not necessary) condition for splittability [13].

Sample populations (relations) can help to reveal the absence of constraints, but they cannot, even in principle, determine whether a relation scheme is atomic. This is because there is no formal way that a sample population can be determined to be significant in this sense. In the final analysis, *the atomicity of a relation scheme depends on what the relation scheme means and on the nature of the business domain*. The only safe way to resolve this is to check with a domain expert who understands both these aspects, and this is necessarily an *informal* process.

ORM includes formal machinery for determining whether one fact type is equivalent to a conjunction of other fact types, but equivalence proofs rely on the provision of contextual definitions for defining predicates in one representation in terms of predicates in the other representation [11, 12], and the provision of such context is an informal process that again relies on a domain expert who understands both the predicate meanings and the business domain. This is a matter of logic that applies regardless of the modeling approach used.



D1:
$\forall x{:}A\ \forall y{:}B\ (xSy \equiv \exists z{:}C\ Rxyz)$
$\forall x{:}A\ \forall y{:}C\ (xTy \equiv \exists z{:}B\ Rxzy)$

D2:
$\forall x{:}A\ \forall y{:}B\ \forall z{:}C\ (Rxyz\ .\equiv.\ xSy\ \&\ xTz)$

Fig. 5. A projection-join equivalence

For example, Fig. 5 depicts one of many projection-join equivalence patterns in ORM. Conceptual schema 1 (CS1) is a ternary fact type, while conceptual schema 2 (CS2) comprises two binary fact types. The equivalence formulae in contextual definition D1 define the predicates of CS2 in terms of the predicates of CS1, and the D2 formula defines the CS1 predicate in terms of CS2's predicates, thus providing conservative extensions to each schema, which allow the equivalence to be formally proved using standard logical techniques such as deduction trees [11].

The relation schemes in Fig. 1 and Fig. 2 respectively conform to the ternary and binary patterns CS1 and CS2. Whether these relation schemes are equivalent depends on whether the semantics of the schemes match the patterns shown in the contextual definitions D1 and D2. Using our first interpretation of the schemes (Person plays Sport and speaks Language), D1 and D2 become (using sorted logic with mixfix predicates):

D1: $\forall x{:}Person\ \forall y{:}Sport$ (x plays $y \equiv \exists z{:}Language\ x$ plays y and speaks z)
$\forall x{:}Person\ \forall y{:}Language$ (x speaks $y \equiv \exists z{:}Sport\ x$ plays z and speaks y)

D2: $\forall x{:}Person\ \forall y{:}Sport\ \forall z{:}Language$ (x plays y and speaks $z\ .\equiv.\ x$ plays $y\ \&\ x$ speaks z)

Although this example is trivial, whether the ternary is decomposable into the two binaries depends totally on whether these definitions apply, and this decision is an informal issue to be decided by the domain expert. ORM provides a sugared textual language to render the equivalences in a form more digestible to nontechnical users, essentially asking whether the ternary can be rephrased as a conjunction of the binaries. As no such definitional context can be provided for the alternative semantics (Person plays Sport only if it is refereed in Language), the decomposition is ruled out, and again this is an informal issue. These decisions can be made merely by understanding the semantics or meaning of the predicates, rather than relying on inspection of sample relations that are possibly not transparent.

From a proof-theoretic perspective, once the domain expert agrees to the conjunction claim (and hence the relevant definitional context), the matter is settled. From a model-theoretic perspective, the equivalence applies if and only if the conservatively extended schemas have exactly the same models (interpretations that are true for the UoD), and this requires agreement between transparent relations. But pragmatically, the model theoretic approach is of little direct, practical use, because in assigning a relation to each predicate (part of the task of providing an interpretation), one tacitly as-

sumes that the relation contains all the true tuples for that population of individuals (i.e. the relation is transparent). But in order to know the relation is transparent, we need to know whether the relation is conjunctive for that business domain.

The essential confusion in the traditional definitions of MVDs etc. hinged on the loose notion of "possible instance" of a relation scheme. In practice, many possible but nontransparent instances may be provided, and the only way to detect problems with these is to return to the fundamental notion of logical conjunction. Recasting the notions of MVDs, 4NF, JDs, and 5NF in terms of logical equivalence involving conjunctions provides the only truly *semantic* formulation of these concepts.

For example, let *R(X, Y, Z)* be an ORM fact type where *R* is the logical predicate and *X*, *Y*, *Z* are role sequences (null, unit, or composite). We say that *R* includes the semantic MVD $X \twoheadrightarrow Y$ if and only if for each possible UoD state, each fact instance $R(x, y, z)$ has the same truth value as that of the conjunction $S(x, y)$ & $T(x, z)$ for some predicates *S* and *T*. This is equivalent to the definition of semantic MVD given earlier, but is more useful as it relates directly to the fundamental equivalence question to be answered by the domain expert in determining atomicity of a fact type. Similarly, semantic JDs may be defined in terms of *n*-term conjunctions ($n > 1$).

Once atomic relation schemes are determined, to get a relational schema all of whose relation schemes are in 5NF, we synthesize new, possibly nonatomic relation schemes from the atomic relation schemes where that can be done without introducing nontrivial scheme-MVDs or -JDs, or any nontrivial FDs not enforceable by key constraints. This synthesis can be done algorithmically, based on key and other constraints, as illustrated by ORM's Rmap algorithm [13].

Thus, we see that the steps which ORM includes pursuant to a fully normalized relational database schema, including conceptualization in terms of atomic facts followed by application of its Rmap procedure, effectively cover not only the formal aspects of normalization theory but also the informal semantic interpretation that is pragmatically needed.

5 Conclusion

This paper identified problems in traditional normalization theory regarding accepted definitions of nonloss decomposition and multivalued and join dependencies (and hence 4NF and 5NF), which unrealistically rely on relations being completely representative. The notion of relation transparency was introduced to refine these definitions, thus providing a theoretical resolution of these issues. However, a pragmatic solution to these problems was seen to require a judgment on conjunction based, semantic equivalences, an essentially informal process involving the understanding of the domain expert rather than inspection of sample relations.

The modeling techniques used in ORM, which begin with establishing atomic fact types, later grouped into relation schemes using a well known mapping algorithm, provide one practical realization of the recommended approach. While this basic approach could be adapted to other modeling approaches such as ER and the Unified Modeling Language (UML), ORM's emphasis on communication in natural language sentences seems to make it especially suitable for this kind of procedure.

Step 1 of ORM's Conceptual Schema Design Procedure begins by having domain experts verbalize concrete information examples of interest in natural language sentences. A later stage of this step requires the modeler to rephrase the information in terms of atomic facts, checking atomicity with the domain expert by asking whether a sentence of the specified kind can be equivalent to a conjunction of smaller sentences (using concrete instances). Once this is established, the modeler abstracts from the fact instances to the fact types. Either now or later in the design procedure, modelers may also draw on ORM theory that clarifies how the presence of uniqueness constraints impacts atomicity. This procedure has time and again proved effective in industrial modeling. By ensuring atomicity at the front end, it is relatively easy to later ensure that fact types are grouped into fully normalized relation schemes.

The current paper strikes at the heart of the procedure promoted by standard normalization theory, inasmuch as it undermines the syntactically-based, too restrictive definition of "nonloss decomposition" that underlies that procedure. Clearly, a more semantics-based "normalization procedure" is required: as we have shown, only a procedure based on the informal semantics, and specifically one that will determine whether that semantics is conjunctive, is adequate to the problem of normalization. The implication for teaching normalization is that both the theory and the procedure taught, must be adjusted to compensate for these problems in the traditional treatments.

It is true that current normalization practice tends to ignore 4NF and 5NF; and some might see that fact as undermining the relevance of our findings about the theory and method of normalization. However, to ignore 4NF and 5NF does not fix the problem in the theory, nor provide an alternative way to arrive at a fully normalized schema. As we stated at the outset, normalization is a good, needful thing; however, a theory that incorrectly states the criteria of "fully normalized", and thereby makes reaching that goal impractical, is a theory that needs amending.

References

1. Aho, A. V., Beeri, C. & Ullman, J. D. 1979, 'The Theory of Joins in Relational Databases', *ACM Transactions on Database Systems*, vol. 4, no. 3. First published in Proc. 19th IEEE Symp. on Foundations of Computer Science (October 1977).
2. Batini, C., Ceri, S. & Navathe, S. 1992, *Conceptual Database Design: an entity-relationship approach*, Benjamin/Cummings, Redwood City.
3. Codd, E. 1970, 'A Relational Model of Data for Large Shared Data Banks', *Communications of the ACM*, vol. 13, no. 6, pp. 377–87.
4. Codd, E. F. 1971, 'Further Normalization of the Data Base Relational Model'. (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems," New York City, May 24th-25th, 1971.) IBM Research Report RJ909. Republished in Rustin, R. J. (ed.), *Data Base Systems: Courant Computer Science Symposia Series* 6, Prentice-Hall, 1972, pp. 33–64.
5. Date, C. J. 2000, *An Introduction to Database Systems*, 7th ed., Addison Wesley Longman.
6. Date, C. J. & Fagin, R. 1992, 'Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases', *ACM Transactions on Database Systems*, vol. 17, no. 3, pp. 465–476.
7. Elmasri, R. & Navathe, S. 1994, *Fundamentals of Database Systems*, 2nd ed., Addison-Wesley.

8. Fagin, R. 1977, 'Multivalued Dependencies and a New Normal Form for Relational Databases', *ACM Transactions on Database Systems*, vol. 2, no. 3, pp. 262–278.
9. Fagin, R. 1977, 'Functional Dependencies in a Relational Database and Propositional Logic', *IBM Journal of Research and Development*, vol. 21, no. 6, pp. 534–544.
10. Fagin, R. & Vardi, M. Y. 1986, 'The Theory of Data Dependencies', *Mathematics of Information Processing*, eds. M. Anshel & G. Gewirtz, Proceedings of Symposia in Applied Mathematics, vol. 34, pp. 19–71, American Mathematical Society, Providence.
11. Halpin, T. 1989, 'A Logical Analysis of Information Systems: Static Aspects of the Data-Oriented Perspective', PhD thesis, University of Queensland.
12. Halpin, T. and Proper, H. 1995, 'Database Schema Transformation and Optimization', *Proc. OOER'95*, ed. M. Papazoglou, Springer LNCS, no. 1021, pp. 191–203.
13. Halpin, T. 2001, *Information Modeling and Relational Databases*, Morgan Kaufmann, San Francisco.
14. Halpin, T. 2006, 'Object-Role Modeling (ORM/NIAM)', *Handbook on Architectures of Information Systems, 2nd edition*, Springer, Heidelberg, pp. 81–103.
15. Halpin, T. 2007, 'Fact-Oriented Modeling: Past, Present and Future', *Conceptual Modelling in Inf. Sys. Eng.*, eds. J. Krogstie, A. Opdahl & S. Brinkkemper, Springer, pp. 19–38.
16. Object Management Group 2008, *Semantics of Business Vocabulary and Business Rules (SBVR), v1.0*. Online at http://www.omg.org/spec/SBVR/1.0/PDF.
17. O'Neil, P. & O'Neil, E. 2001, *Database Principles, Programming, and Performance*, 2nd ed., Morgan Kaufmann Publishers, San Francisco.
18. Ritson, P. and Halpin, T. 1993, 'Mapping Integrity Constraints to a Relational Schema', *Proc. 4th ACIS*, Brisbane (Sep.), pp. 381–400.
19. Simsion, G. & Witt, G. 2005, *Data Modeling Essentials*, 3rd ed., Morgan Kaufmann Publishers.
20. Teorey, T. 1999, *Database Modeling and Design*, 3rd ed., Morgan Kaufmann Publishers.
21. ter Hofstede, A., Proper, H. and van der Weide T. 1993, 'Formal Definition of a Conceptual Language for the Description and Manipulation of Information Models', *Information Systems*, vol. 18, no. 7, pp. 489–523.
22. van Griethuysen, J. (ed.) 1982, *Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO TC97/SC5/WG3, Eindhoven.
23. Verheijen, G. and van Bekkum, J. 1982, 'NIAM: An Information Analysis Method', *Information systems Design Methodologies: a comparative review, Proc. IFIP WG8.1 Working Conf.*, Noordwijkerhout, The Netherlands, North Holland Publishing, pp. 537–90.