Teaching Tip

Clarifying Normalization

Donald A. Carpenter
Department of Business
Mesa State College
Grand Junction, CO 81501 USA
dcarpent@mesastate.edu

ABSTRACT

Confusion exists among database textbooks as to the goal of normalization as well as to which normal form a designer should aspire. This article discusses such discrepancies with the intention of simplifying normalization for both teacher and student. This author's industry and classroom experiences indicate such simplification yields quicker learning and more complete understanding by students.

Keywords: Normalization, Normal Forms, Database, Data Redundancy, Structure Redundancy

1. INTRODUCTION

A perusal of database textbooks exposes an obvious disagreement as to how to teach database normalization. Apparently, there are divided opinions as to what normalization is supposed to accomplish. Similarly, there are conflicting statements regarding which normal form (NF) a database designer should aspire to reach. This paper attempts to bring some resolution to both issues, in order to improve the teaching and comprehension of database normalization.

Conversely, it is not the intention of this paper to explain database normalization in detail. However, it is this author's belief that clarification of the two above-noted issues will improve the understanding of the nature of normalization to the reader who does not have a strong grasp of the topic. For readers' reference, Table 1 lists the normal forms that are presently known. Included is the zero normal form (0NF), a device this author uses to help explain the milieu to students. A table can be in the zero normal form but, by definition, a relation cannot.

2. WHAT IS NORMALIZATION SUPPOSED TO ACCOMPLISH?

E. F. Codd, the acknowledged father of relational database and normalization, initially defined normalization as the "very simple elimination procedure" to remove non-simple domains from relations (Codd, 1970, p. 381). A relation with simple domains is one whose elements are atomic or non-decomposable (p. 380). In other words, the attributes in a relation with simple domains are properly tied (i.e., dependent upon) only to the attribute(s) that uniquely identifies all others (i.e., the primary key). Hoffer et al., (2007) state it well enough as "Normalization is the process of successively reducing relations with anomalies to produce small, well-structured relations" (p. 211). The reader should note that "small" is a relative term. A properly normalized relation with a couple dozen attributes is not as small as an unnormalized relation with a half-dozen attributes. Both large and small relations are a part of the reality in corporate information structures.

Normal Form	Defining Status
0NF	A table that has not been normalized, i.e., not proven to be a true relation.
1NF	A true relation (i.e., no repeating groups of attributes and each row is unique).
2NF	A relation in 1NF + all non-key attributes are dependent on all of the primary key.
3NF	A relation in 2NF + all non-key attributes are dependent only on the primary key.
BCNF*	A relation in 3NF + there is only one primary key within the relation.
4NF	A relation in BCNF + no multi-valued dependencies.
5NF, a.k.a. PJNF**	A relation in 4NF + any join dependencies are based on candidate keys.
6NF	A relation in 5NF + no non-trivial join dependencies, including temporal data.
DKNF***	A relation in which all constraints are functions of domains and primary keys.

* Boyce-Codd Normal Form ** Project-Join Normal Form *** Domain-Key Normal Form

Table 1. Normal Forms

However, Hoffer et al. (2007) go on to list as "some of the main goals of normalization:

1. Minimize data redundancy, thereby avoiding anomalies and conserving storage space.
2. Simplify the enforcement of referential integrity constraints
3. Make is easier to maintain data (insert, update, and delete)
4. Provide a better design that is an improved representation of the real world and a stronger basis for future growth" (p. 211).

Indeed, those might be outcomes and benefits, ones that are achieved to varying degrees from one enterprise information structure to the next. However, the real goal of normalization remains as it was in 1970, to produce correctly structured relations.

It is not this author's intention to single out any author for criticism. Rather, the hope is to clarify the purpose of normalization, as lack of understanding of the goal often leads to students' lack of appreciation of the power of the technique. It also can cause designers to turn to other methods to achieve those goals. Still, there are others whose incorrect statements add to the learning challenge.

For example, one textbook says "Normalization is the process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies" (Rob & Coronel, 2007, p. 148). First, we analyze relations rather than "tables" as a table doesn't qualify to be a relation unless it is shown to be in at least the first normal form. Second, we aim to eliminate modification anomalies rather than reducing "data anomalies." Those two are admittedly somewhat nitpicky objections.

Third and more problematic, though, is the notion that normalization is intended to "minimize data redundancies." Codd (1970) noted that "Redundancy in the named set of relations must be distinguished from the redundancy in the stored set of representations. We are primarily concerned here with the former" (p. 385). It is the latter that is properly referred to as "data redundancies," while the former could be better called "structure redundancies."

Moreover, Codd (1970) went on to describe both strong and weak redundancies. Again, it is not the purpose of this paper to thoroughly explain those. Suffice it to again quote Codd about the significance of each of those. "An important reason for the existence of strong redundancies in the named set of relationships is user convenience" (p. 386). "Generally speaking, weak redundancies are inherent in the logical needs of the community of users. They are not removable by the system or data base administrator" (p. 386).

Thus, a certain amount of structure redundancy is desirable. Normalization, if properly applied to relations, will assure that structure redundancies are properly stated. A common example of structure redundancy is the placement of a foreign key into a relation. Since a foreign key is a primary key and an attribute in some other relation, the placement of a foreign key creates redundancy in the overall database structure.

Thorough normalization also can minimize the chances that unnecessary data redundancy might occur. Typically, however, protection against data redundancy is enforced via integrity controls imposed when relations are implemented physically. Such protection notwithstanding, data redundancies can be part of the physical design even in a highly normalized set of relations. An example of beneficial data redundancy results with replicated data bases, when data is deliberately copied onto multiple platforms.

It may well be a natural tendency of textbook authors to expand on a concept to differentiate the coverage in one book from its competitors. However, there are times when it is best to stick with the simple historical definition. While normalization might result in many more advantages, it would be best for students if authors stuck to the sole goal of normalization: to produce properly structured relations.

3. HOW FAR SHOULD ONE NORMALIZE?

Normal forms provide check points for normalization. Nine normal forms are listed in Table 1. This author included his own 0NF, which he has not found in print elsewhere. However, not all textbook authors note the existence of the other eight. Moreover, there is disagreement as to the point where normalization should end.

Rob & Coronel (2007) take the stance that "Almost all business designs use 3NF as the ideal normal form" (p. 173). Also, "Tables in 3NF will perform suitably in business transactional databases. However, there are occasions when higher normal forms are useful" (p. 163). Those authors do go on to illustrate BKNF and 4NF.

Similarly, Hoffer et al. (2007) state "Relationships in third normal form (3NF) are sufficient for most practical database applications" (p. 572). However, they do explain BKNF and 4NF, although those are relegated to an appendix. That textbook does not explain 5NF and 6NF.

Pratt and Adamski (2005) state "The most common normal forms are the first normal form (1NF), second normal form (2NF), third normal form (3NF), and fourth normal form (4NF)" (p. 140). They do not make a distinction between the BKNF and 3NF but use the definition of BKNF as the definition of 3NF (p. 153). They do not discuss application of normal forms above BKNF, but suggest that one can avoid the problem of multivalued dependencies (4NF problems) by use of a design methodology (p. 166).

According to Mannino (2004), "The normalization story does not end with 4NF. Other normal forms have been proposed, but their practicality has not been demonstrated" (p 245). Consequently, he demonstrates normal forms only through 4NF.

Ulman & Widom (2008) set a different approach by discussing only the Boyce-Codd Normal Form. "There is, it turns out, a simple condition under which the anomalies discussed above can be guaranteed not to exist. This condition is called *Boyce-Codd normal form*, or *BCNF*" (p. 88).

Kroenke (2006) takes an approach similar to Ulman & Widom, although he goes one normal form higher in his illustrations. First, though, he groups the normal forms into three levels (see Table 2). Then he explains that one can overcome all the problems addressed by the lower normal forms simply by normalizing to 4NF. He also explains 5NF and DKNF by stating "The third source of anomalies is esoteric. These problems involve specific, rare, and even strange data constraints" (p. 83).

Source of Anomaly	**Normal Forms**	**Design Principles**
Functional Dependencies	1NF, 2NF, 3NF, BCNF	BCNF: Design tables so that every determinant is a candidate key
Multivalued Dependencies	4NF	4NF: Move each multivalued dependency to a table of its own
Data constraints and oddities	5NF, DK/NF	DK/NF: Make every constraint a logical consequence candidate keys and domains.

Table 2. Groupings of Normal Forms From Kroenke (2006, p. 83)

The disagreement among the authorities as to which normal form is far enough is confusing for both students and teachers. The concept put forth by both Ulman & Widom and Kroenke that a database designer simply can use one normal form (BCNF or 4NF, respectively) to eliminate all lower level problems is attractive from a teaching perspective. Certainly teaching one normal form could be easier and less time consuming than teaching four or five. Indeed, this author advocates the Kroenke approach as classroom experience indicates faster learning and a deeper levels of understanding.

But how does one teach normalization by using only one normal form? As stated previously, it is not the intention of this paper to teach normalization. Nor is it an intention to teach how to teach normalization. However, this author finds it useful to instruct students to follow the wisdom of "the anonymous ditty found in Kroenke (2006): 'I swear to construct my [relations] so that all non-key [attributes] are dependent on the key, the whole key, and nothing but the key, so help me Codd'" (p. 88).

However, the question remains "Why not move on to even higher normal forms?" Are the higher normal forms impractical or esoteric, as suggested in textbooks? Should it be sufficient to leave students with the impression that 3NF relations are suitable for business or sufficient for most applications? Whereas many installed databases might indeed wind up with relations mostly in 3NF, it is misleading to tell students that it is okay to stop with 3NF.

There is a better business-oriented approach to the issue. Figure 1 presents a stylized column chart based on this author's numerous experiences with live corporate information structures. First, the horizontal axis shows that reaching each normal form requires its own increment of time. Experience with dozens of clients says, for example, that reaching 5NF requires substantially more time than reaching each of the other normal forms. The vertical axis illustrates the relative amount of future anomalies removed by moving to each higher normal form. The amount removed by reaching 5NF is very low by comparison to the success level of reaching the lower normal forms.

What that chart shows is a simple business return on investments model. The return (the reduction in future anomalies) on investment (time spent on normalizing) from striving for 5NF is simply not as great as it is for lower normal forms. The problem of stopping short of achieving higher normal forms is that the anomalies associated with those normal forms are not removed. The good news is that there is a different solution. If such 5NF anomalies are encountered at a later date, software routines can be written to work around them.

Not normalizing to 6NF is a different matter than not normalizing to 5NF. Date, et al., (2002) explain 6NF as pertaining to temporal data. In this author's experience, temporal data with 6NF-type anomalies are much more common in data warehousing installations than in databases used primarily for daily operations. Therefore, the return on

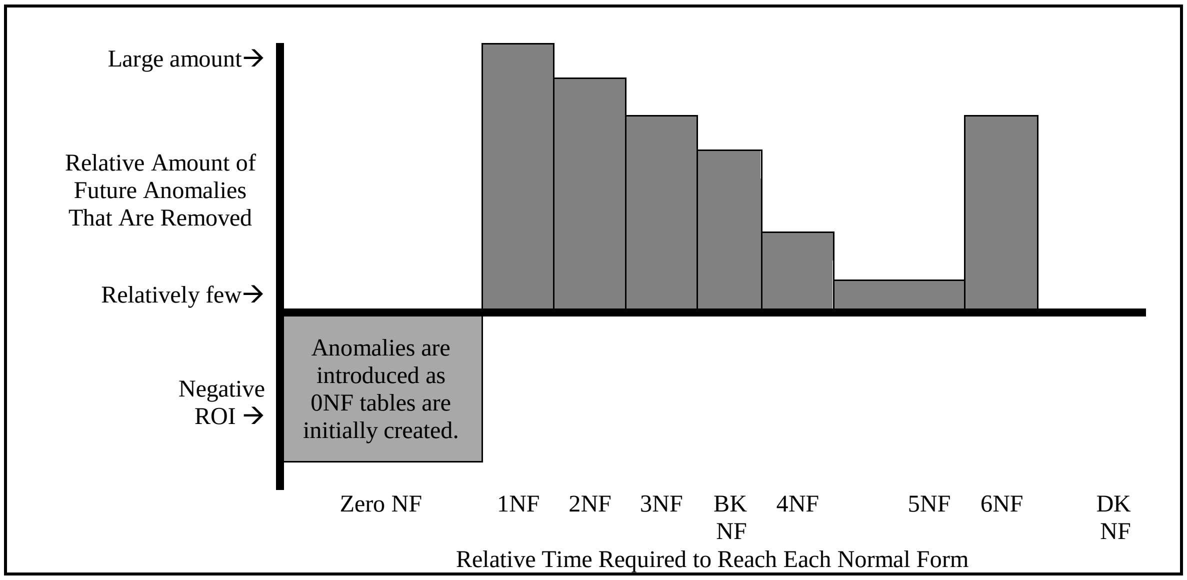

Figure 1. Generalized Return on Investment of Normalization

investment might not be justified unless the firm is designing a data warehouse. The good news here is that 6NF normalization can be deferred to the time when such a data warehouse is contemplated, in this author's experience.

Failure to normalize to DKNF is an oxymoron. No one has experience with DKNF as it was a straw dog attempt to define perfection without giving any method for getting there or any precise description of what it looks like once one is there. If a database designer was to spend time trying to normalize to DKNF, the return on investment is predictably zero.

On the left end of the scale, the return (amount of future anomalies removed) on investment (time) of reaching 0NF is zero at best, but most likely negative. That is the nature of 0NF relations, which are the outcomes of the logical design stage in which tables (not relations) are initially formed and, therefore, the stage when future anomalies are introduced into the database structure. Hence, one could argue that the ROI is actually negative.

It is important to deal with a related issue. Normalization typically results in more and smaller relations. A substantial number of small relations can result in lack of efficiency within hardware storage subsystems, more software overhead, lack of convenience for users, and lack of managerial simplicity for database administrators.

To address that situation, of course, there is the solution of denormalization. However, that should be held in reserve until after the database designers have proven the database structure to be as sound as possible. Such denormalization should be applied judiciously.

If denormalization is used, does that mean time has been wasted doing normalization? Not if normalization is considered an insurance policy. It is better to remove as many future anomalies as possible first. Then, reintroducing potential problems by means of denormalization gives data administrators a clear indication of exactly where those future anomalies might occur.

4. SUMMARY

This article has pointed out the discrepancies among textbooks regarding two important concepts of normalization. First, the goal of normalization should be standardized back to Codd's original definition. The many potential outcomes and benefits of normalization should not be confused with the overriding goal, which is to create properly structured relations, as those contain fewer future anomalies. Focusing on benefits instead of the goal can mislead one to bypass normalization altogether since there are other ways to at least partially achieve some of those same benefits that are misstated as being goals.

The question of how far should a database be normalized has been addressed by use of a simple return on investment model. Stopping short of 4NF is not a wise business decision as many anomalies might still exist. A designer should reach the fourth normal form. Moving beyond the 4NF should be examined in terms of ROI. In this author's experiences, striving for 5NF seldom produces significant results and typically is not justified based on ROI. However, the sixth normal form should be reached if temporal data is prevalent, as in a data warehouse, which requires achieving 5NF first.

5. REFERENCES

Codd, E. F. (1970). "A relational model of data for large shared data banks." Communications of the ACM, Vol. 13 (6). 377-387.

Date, C. J., Darwen, H., & Lorentzos, N. A. (2002). Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and Relation Theory to the Problem of Temporal Database Management. Amsterdam: Elsevier Science Ltd.

Hoffer, J. A., Prescott, M. B., & McFadden, F. R. (2007). Modern Database Management, 8th ed. Upper Saddle River, NJ: Pearson Prentice Hall.

Kroenke, D. M. (2006). Database Processing: Fundamentals, Design, and Implementation, 10th ed. Upper Saddle River, NJ: Pearson Prentice Hall.

Mannino, M. V. (2004). Database Design, Application Development and Administration. Boston: McGraw Hill Irwin.

Pratt, P. J., & Adamski, J. J. (2005). Concepts of Database Management, 5th ed. Boston: Thompson Course Technology.

Rob, P., & Coronel, C. (2007). Database Systems: Design, Implementation, and Management, 7th ed. Boston: Thompson Course Technology.

Ulman, J. D., & Widom, J. (2008). A First Course in Database Systems, 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall.

AUTHOR BIOGRAPHY

Donald A. Carpenter is a professor in the Department of Business at Mesa State College, primarily teaching computer information systems courses. He earned a Ph.D. in Management Information Systems at the University of Nebraska-Lincoln, an MBA in Information Systems at the University of Colorado in Colorado Springs and a Bachelor of Science in Business Administration at Kearney State College. He spent ten years marketing large systems in the computer industry prior to becoming a college professor. He has consulted extensively since then and has been involved with the analysis, design and implementation of over 200 corporate databases. He is published in a variety of information systems journals and conference proceedings. His research interests are in information requirements determination, information systems pedagogy, and program assessment.

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096