

## **Modeling Matters: Logical Data Models, Denormalization and Conceptual Clarification: Part 2.**

Dr. Tom Johnston  
Mindful Data, Inc.

Originally published at DataWarehouse.com in July, 2002.  
Reprinted here by permission of the senior editor at DM Review.

### ***Journeyman Data Modeling.***

Journeyman data modelers, among whom I must count the majority of modelers I have met, do not use the exercise of creating models to engage the user community in conceptual clarification. Instead, they build a data model entirely on the concepts, and especially the ontology, that the user community is familiar with.

In the first scenario, for example – the one about contracts and contract versions – the easiest course of action is to accept the ontology of the business as it exists, and to include in the model a Contract entity whose business identifier is contract-number plus contract-renewal-date.

In the second scenario, the data modeler reaches the point where he realizes that the minimum cardinality of the relationship – whether a customer can exist with no salesperson, and vice versa – has not been specified in the requirements. Instead of asking for clarification, however, it is easiest to assume the most common case, i.e. that the relationship is optional for both customers and salespersons. This case permits all valid data to be recorded in the database, no matter what the real rules about minimum cardinality are.

In both cases, the completed model will direct the construction of a database that does work. In the former case, the business has continued to operate for years or perhaps decades with databases that did not distinguish contract versions from contracts, even though the users were often interested in all the contract versions with the same contract number, i.e. in contracts. Certainly, then, the business can continue to operate that way. Indeed, as I indicated in the previous installment in this series, even very, very bad data

models can be implemented in information systems that still meet all the user requirements.

In the latter case, by making the relationship optional for both customers and salespersons, the database is still capable of *expressing* the true business policy, even if that policy is that a salesperson is required for a customer, or vice versa, or both. What has been lost is the ability to *enforce* those constraints by the DBMS itself. Since those constraints were not specified in the requirements, it is likely that they will not be enforced in code, either. This leaves their enforcement to the mechanism of last resort – user convention. If users who enter customers into the Customer table, and salespersons into the Salesperson table, are careful enough, it may be a long time before a salesperson-less customer is recorded in the database, or a customer-less salesperson.

This kind of journeyman data modeling, as I said above, does work. It is easy for a data modeler to do only this kind of work, for any of several reasons. In some cases, the modeler doesn't realize that with an adjustment to the users' conceptual scheme, a lot of convoluted business processes could be simplified. With adjustment to the users' ontology, as reflected in new entities being introduced into the data model, the code that accesses and maintains that database can be substantially simpler, and the database itself will be easier to understand once a period of time has elapsed to allow the users to accommodate themselves to the conceptual changes.

In other cases, data modeling is relegated to a back-room function. Someone else, often someone unfamiliar with data modeling, gathers user requirements. At the end of the requirements definition phase, the requirements are turned over to the data modeler. As the model is constructed, the modeler will almost inevitably attempt to draw relationships whose full cardinalities are not specified in the requirements. For example, he might find an entity involved in a set of relationships each of which is optional for that entity, but suspect that at least one of them is required for every instance of the entity. Once again, data-related business rules like these are often not captured in requirements gathering sessions. But by this time, requirements definition is over. A few of the most serious issues the

modeler discovers may get answered, but the channels to the subject matter experts are no longer wide open.

### ***Professional Data Modeling.***

Professional data modelers have an obligation to create the best data models they can for their employers. The first scenario illustrated a case where doing so requires the modeler to persuade the business experts to change their conceptual scheme. This is often a very difficult thing to do; but the responsibility does exist. The second scenario illustrated a case where developing the data model raised business requirements questions that were not answered during the formal requirements definition process. In this case, it is easier, and quicker, to make a guess than to go back to the subject matter experts and clarify the matter. Far too many business rules are defined by IT personnel, in the process of making guesses like these. Some of them are expressed in data schemas, others in procedural code. It should go without saying that it is not the role of a modeler, DBA or developer to define business rules.

The discovery of as-yet undefined business rules is what I consider the major benefit of creating normalized data models. They supplement the passive activity of writing down whatever requirements the subject matter experts think to tell us with an active process of discovering data-related requirements that were overlooked. In this respect, data modeling is better at bringing out all relevant requirements than are use cases, because building a logical model brings out requirements issues that may not have been raised. The passive process of writing use cases, on the other hand, no matter how smart and interactive the business analyst is, provides no mechanism by which to discover gaps in the requirements. The mechanics of normalization force the discovery of data requirements, instead of passively relying on subject matter experts to remember to tell us about them.

Data modeling beyond the journeyman stage, then, is much more than making entities out of the noun phrases and relationships out of the verb phrases in requirements documents. It is an active and interactive process of engaging subject matter experts in the clarification of essential business concepts, often concepts so central that they reflect the underlying ontology

of the business. The value of such clarification is that while poor data models can usually be made to work, systems based on clarifying and improving the conceptual scheme of the business will be more reliable, flexible and efficient in the expression and enforcement of the applicable business rules. A second way in which professional data modeling is an active process is that the construction of a data model raises requirements questions that would otherwise have been overlooked. To an inexperienced modeler, these look like technical questions of business identifiers and relationship cardinalities. To the professional modeler, it is apparent that these are questions about the very meaning of the most fundamental concepts of the business.

In this way, professional data modeling leads to a true dialog with user experts in which the modeler is given the means to "push back" and challenge what the user experts say. Time and time again, I have found that in doing so, those user experts are capable of looking at their business in a new way. The result, in such cases, is more than a workable model, a workable database and a workable system. It is a new way of thinking about the business, one which results in more reliable, flexible and efficient information systems, and more streamlined business processes.

### ***The Enforcement of Integrity Constraints: Another Point of View.***

As I was submitting this article, I came across the first of a two-part series which, just as Dr. Sharp's article, seems to oppose the position I have been arguing for.<sup>1</sup> In that article, Fabian Pascal says that he will explain "just why the notion of 'denormalization for performance' is a misconception", and will "expose its costly dangers, of which practitioners are blissfully unaware. If the integrity cost of denormalization is taken into consideration," he continues, "it will cancel out performance gains, if any."

Although I write this prior to the publication of the second installment in Pascal's series, the thrust of his argument, if not all the details, seems clear.

---

<sup>1</sup> "The Dangerous Illusion: Denormalization, Performance and Integrity, Part 1", Fabian Pascal. *DM Review*, June 2002.

What he calls “integrity” issues are what I have called “business concepts” which, as a totality, constitute the “conceptual scheme” for a business, variously expressed in business rules, policies and other constraints”.

I have argued that a fully-normalized logical data model brings to light most of the data-related business rules for the data being modeled. My response to Pascal’s assertion that “the integrity cost of denormalization . . . . will cancel out performance gains”, as well as to Dr. Sharp’s assertion that “Any structure that . . . . allows invalid data to be populated must be considered an error”, is that the key to data integrity is to *identify* all the relevant business rules, not necessarily to *enforce* those rules in one specific way, i.e. by means of fully normalized physical data schemas.

If we fail to identify a business rule, the integrity of the data it governs will not be enforced because we won’t know to do it. But if we do identify the rule, then we have several ways to enforce it and insure the integrity of the data it governs. Physically implementing a fully-normalized data model is only one of the ways to do that. If the cost of such a database is unacceptable performance, then let us denormalize, achieve the performance the user demands, and create developer-written code to enforce the business rules which are no longer enforced by data structures.<sup>2</sup>

Just as firmly as do Pascal and Dr. Sharp, I insist that the integrity of the data must not be compromised. But I also insist that the ultimate objective is to satisfy the business user. As long as the process of developing a normalized model has brought integrity constraints to light, we can then chose among various means of enforcing those constraints. Denormalization often moves the enforcement burden to developer-written code. But if, in the process, we bring the performance of our databases to an acceptable level, then we have satisfied the business user in terms both of performance and of the integrity of the data.

The cost of enforcing integrity constraints by means of developer-written code is indeed higher than the cost of enforcing it by means of data

---

<sup>2</sup> My nine-part series on “Basic Normalization”, especially the later installments which discuss Boyce-Codd normal form, also contain a discussion of the various ways that data-related business rules can be enforced. This series can be found in the archives on this site.

structures and the DBMS engine. One of those costs is that code is more error-prone than data structures. One reason is that the code is a procedural specification of an integrity constraint while data structures are a declarative specification. But if acceptable performance can be achieved in no other way, and acceptable performance is a requirement, then the additional cost may often be worth the result it achieves – a satisfied user.

Nonetheless, I encourage my readers to read Dr. Sharp's article, and to examine Pascal's two articles very carefully. Their continued emphasis on the value of relational theory to practical issues of implementing databases, is very important. As Chris Date has repeatedly said, "Theory is practical!", and I agree completely.

The most useful work we data modelers can do lies at the intersection of theory and practice, but I believe that neither can dominate the other. I believe that no set of non-trivial rules contains a complete set of instructions for applying itself to every conceivable situation. Aristotle noted an analogous gap in ethics, a gap between the most precisely-specified ethical rules, and the day-to-day situations they apply to. For Aristotle, that gap was filled by *praxis*, a faculty of moral judgment which was governed by experience and moral maturity, not by rules. Similarly, modeling decisions made on the grounds of theoretical purity are sub-optimal; they optimize the enforcement of data integrity rules without code, but often at the cost of failing to optimize the business user's satisfaction with the complete system. In the application of modeling theory to modeling practice, we need a faculty of data modeling *praxis*. If we simply slam theoretical purity onto every relational modeling situation we are involved in, we are optimizing the value of something we care about at the cost of something the business user cares about.