

Is Codd Dead?

Dawn
(TinCat Group)

Time Magazine issued a cover in 1966 asking "Is God Dead?" This is not the first time the name "Codd" has replaced "God" in a *phrase*, but in this case it is not for the purpose of comparison. In this blog, I will dare to question some of Codd's legacy including some of the dogma passed along in college database textbooks today.



E.F. (Ted) Codd died in 2003 leaving a significant contribution. Codd is often called the father of relational theory. His 1970 ACM paper *A Relational Model of Data for Large Shared Data Banks* (E. F. Codd, Communications of the ACM, v.13 n.6, p.377-387, June 1970) is a significant industry milestone.

In this paper, Codd discusses what he sees as the advantages in modeling *data* by use of mathematical *relations* compared to mathematical graphs of trees or networks.

Relations are often represented as tables of rows and columns. Trees are often visualized as nested folders and documents. The network graph, seen by Codd as overly complex and a cause of some of the problems he was addressing, can be visualized as a web. While a web, or directed-graph, might be a more complex mathematical structure than a relation, I predict that this data model might just catch on anyway (wink).

There are many viable models for data. Each has its advantages and disadvantages. This blog will not be about right and wrong as much as better and worse approaches. I'm a practitioner dabbling in theory in order to help improve the practice and not the other way around.

My advice is this: Stop normalizing your data. Stop removing all repeating groups.

Codd also introduces the term "normalize" to refer to removing *nonsimple domains*, such as lists or tables of data often referred to as "repeating groups." He is very clear in this paper that a relation could include repeating groups, but that normalizing it would make the data model simpler for some purposes.

The simplicity of the array representation which becomes feasible when all relations are cast in normal form is not only an advantage for storage purposes but also for communication of bulk data between systems which use widely different representations of data. (Codd, p. 381)

Is Codd Dead?

Dawn
(TinCat Group)

Anyone communicating bulk data by way of XML or JSON will recognize that we have different issues to solve today than we had in 1970. The rise of XML with its associated unnormalized data model is part of the impetus for what will likely be significant changes on the database landscape.

My advice is this: Stop normalizing your data. Stop removing all repeating groups. Note that I am using the original description of normalization from this paper. This meaning of normalization was later termed, or at least rolled into the term, "First Normal Form" or 1NF. The higher normal forms, such as BCNF, include laudable work with functional dependencies, but all are defined to first require normalization. There is definitely some good that can be salvaged from this normalizing debacle of the past few decades, but we must first ditch the requirement for data to be normalized, placed in 1NF, stripped of repeating groups. I will refer to relations that are not normalized, as others have, as NF2 for Non-First Normal Form.

Do this →

Id: 123456
First: Jayne
Last: VanDoe
Email: jvdoe@abc123.com
 jov@xyz123.com
 jo3@aol.com

Not this →

Id: 123456
First: Jayne
Last: VanDoe

Id: 123456
Email: jvdoe@abc123.com

Id: 123456
Email: jov@xyz123.com

Id: 123456
Email: jo3@aol.com

In this way you will model entities, such as the person above, with their dependent properties, such as the list of e-mail addresses. You only need to remove lists from your model, thereby going from the first example to the second above, if you are using tools that require it. Given that SQL-92 requires it, that is a big if. There are other viable, time-tested NF2 options, however.

Don't be fooled— there is no mathematical requirement to normalize data.

But the Relational Model (RM) is based on mathematics, right? Mathematics is precise. What part of the argument for the RM is amiss? Don't be fooled—there is no mathematical requirement to normalize data. Mathematics provides a means for modeling propositions to be handled in software, presented to end-users, passed as messages, or stored on secondary storage devices. The RM is a mathematical model. It is a model. Models are not the real thing. Models are often anorexic versions of the real thing. The mathematics of the relational model is sound, but the process of determining what this model should be used for is flawed.

Is Codd Dead?

Dawn
(TinCat Group)

The RM has been useful, but not as useful as some pre-relational models, in my opinion. Post-relational models of data for messages, such as those mentioned above, look very much like pre-relational models. I am hoping for a return to best practices for data models, whether or not the theory keeps up. I would, of course, prefer that theory be better aligned with excellent practices. Many pre- and post-relational tools use an NF2 model.

You are likely familiar with RDBMS products, often referred to as relational databases. Purists might prefer these be called SQL-DBMS products since SQL does not promote a pure relational model. I will use this column to dispel what I think to be myths that have helped SQL and the relational model rise to become king of the hill for a couple of decades. While this introductory column is admittedly not meaty, I will delve into this further and provide working examples in the coming weeks.

While I have not yet experimented with any XML DBMS tools, I have been working with one NF2 model, often referred to as the MultiValue (MV) or Pick® data model, for over a decade. This is not the only such model, but one with which I am comfortable, so I will introduce it here and use it in future illustrations and implementations.

Putting the RM and MV side-by-side while wearing both a technical and business woman's hat is what prompted me into further exploration of why the MV data model seems to yield higher productivity for developers, greater flexibility for changes over time, and lower risk of project failure. This was particularly perplexing when I started researching the topic because the RM was developed to help improve database maintenance. While the RM addresses some maintainability issues better than MV, MV seems more flexible in many respects. There are different risks and benefits associated with each approach.

Products employing an MV or NF2 data model include the IBM U2 products, Temenos jBASE, Revelation OpenInsight, Raining Data D3, Northgate Reality, EDP Plc UniVision, Ladybridge Systems OpenQM, and InterSystems Caché. There are other viable functional data model implementations with which I am less familiar, such as Berkeley DB from Sleepycat Software and other products marketed as embedded databases. This is definitely not a small niche market.

OpenQM is an open source implementation, so I will use that for my examples in future blogs. I will be the first to admit that MV isn't new, and although various flavors have tools to make it prettier, it typically doesn't look new. It is unlikely to wow you at first glance, but it often grows on developers quickly with its big bang for the buck results and maintainability. The same principles can be applied to many environments, however, and will typically not be specific to MV tools.

I would like to see the industry start with an NF2 model and move it forward rather than squeeze more out of SQL, as has been attempted with the more recent SQL standards. SQL will be with us for many years, but it is time to make an abrupt cut away from it wherever feasible.

Codd will long be remembered for some very innovative work in the area of database theory. But, yes folks, Codd is dead.