A Solution to the Fatal Flaw in Relational Database Systems

gerryrzeppa Uncategorized May 1, 2018 4 Minutes

Okay, folks, true story.

Way back in the late 1970's I was working as a technical programmer in the database group at Kmart Corporation's world headquarters. IBM's IMS was the mainframe database of choice in those days. But after reading Dr. Edgar F. Codd's seminal paper, *A Relational Model of Data for Large Shared Data Banks*, I was convinced there was a better way.

So over the next couple of years I developed a database design methodology called *Extended Relational Analysis*, and in 1980 I started Relational Systems Corporation to promote it. I had close contacts with the Oracle folks back then (when Oracle employed only six programmers), and I used to demonstrate the virtues of the relational approach in my classes using Oracle via a 300-baud dial-up connection to a VAX-11/780 located in Canada. The folks at General Motors were impressed enough to welcome the Oracle folks into their data centers, and both Oracle and General Motors (and Ford, Boeing, Teradata, EDS, and a number of other big companies) thanked me for my efforts by purchasing licenses to use my training materials. The salad days. Lots of fun, lots of money.

Twenty-nine years later, in 2009, Relational Systems Corporation went bankrupt. Seems I was too busy teaching and doing research to notice that I was spending myself into a hole that I couldn't dig myself out of. Fortunately, it was only money that was lost. The fruits of those fecund decades survive to this day.

Here's what happened. We started out running our business with a mishmash of applications on Macintosh computers. As we grew, two things appeared obvious: (1) we needed a better and more tightly integrated system, and (2) 90% of the desktop computers in use were now Windows machines. So some of my associates and I decided to convert all of our Macintosh stuff (programs and data) to something better on Windows. And then came the surprise (which I report to my shame). When we tried to design a system for our own house — using the techniques we had been so successful at selling to others — we found both the tools and the resulting system to be exceptionally cumbersome and, in the end, altogether untenable.

We had a lot of different *kinds* of data, you see. The usual customer files, employee records, accounting and payroll sheets, of course; but we also had training materials, advertising brochures, and class schedules, together with student registration forms, confirmations, and evaluations; not to mention maps to our classroom and signs for our walls. Now some of those things could be easily

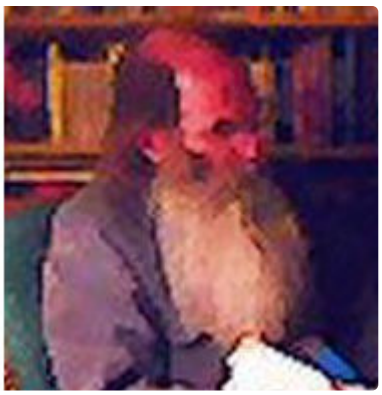
stored and maintained in table format, but much of it just didn't fit. So the fatal flaw that we discovered is that *tables are good for many things, but when you try to get flowers to grow in tables that are not native to that soil, you end up spending way to much time and energy converting the information back and forth from it's normalized storage form to a form the user actually wants to see and use.* After all, an entry form on a screen is not the same thing as a normalized set of tables, and most summary reports don't look like tables, either. Besides, SQL is hardly the weapon of choice when one sits down to tackle a new set of illustrated training materials.

Fortunately, I came across a book by a Russian computer scientist, Mikhail M. Gilula, who was also looking for ways to improve on Dr. Codd's remarkable work. The book was called *The Set Model for Database and Information Systems*, and this then-obscure Russian showed me how a table can hold, not just one kind of row, but many kinds of rows at the same time. And it was then that it dawned on me that what I was looking for wasn't a database, but a *page*base: a collection of Gilula-style sets, but WYSIWYG, with all the virtues of the relational model (like simplicity and closure), and none of the shortfalls. Talk about collusion with the Ruskies! Quick, somebody call CNN!

So my elder son, another associate, and I sat down with Delphi Pascal version 2 and a Windows 95 machine to make this pagebase dream a reality. We wanted tables to become folders, rows to become pages, and columns to become named shapes (ie, fields) on our pages. Ala Gilula, we wanted to allow different *kinds* of pages in each folder. And to maintain relational-style closure (where each operation is "tables in, tables out") we wanted each of the operations of our built-in programming language to be "folders in, folders out."

25,000 lines of code later, it was done. And just two weeks after that, our entire business was up and running on it. We called the program Perspective, and we still use it today, *without modification,* some 20 years later. If you're wondering exactly what a pagebase looks like, take a look at the *Perspective User Manual.* Everything users *and* programmers need to know in just 104 pages! The manual was written and illustrated entirely in Perspective, and thus is just another folder (table) full of pages (rows) in our pagebase.

This is me, then and now…

…and I don't mind telling you that the only thing I regret is not buying Oracle stock.

PS. We never got around to converting our ERA materials to Perspective, but here's a link to our popular *SQL Training Manual*, written entirely in Perspective: each page a "row," with the whole folder a "table" called *SQL Training Manual*. Try to do *that* with nothing but a relational database!