NO DB

Uncle Bob / 15 May 2012 Architecture Tools Craftsmanship Consulting

In the United States, in 1920, the manufacture, sale, and importation of alcoholic beverages was prohibited by a constitutional amendment. That amendment was repealed thirteen years later. During that period of prohibition, the beer industry died.

In 1933, when prohibition was lifted, a few giant grain companies started brewing beer. They completely cornered the market. And for nearly 50 years, we in the United State drank this fizzy bodily effluent and called it "beer". The only way to tolerate the flavor was to drink it very cold.

As a teenager in the '60s, I never understood the attraction. Why beer? It was a pale, yellow, distasteful fluid derived from the urine of sick boars, and had no redeeming qualities that I could see.

In 1984, I went to England; and the scales dropped from my eyes. At last I understood. I had tasted beer for the first time; and I found it to be good.

Since those days the beer situation in the United States has improved dramatically. New beer companies are springing up all over the country; and in many cases the beer they make is actually quite good. We don't have anything quite so nice as a good english bitter; but we're getting close.

In the '80s a few giant database companies cornered the market. They did this by promulgating fear, uncertainty, and doubt amongst managers and marketing people. The word "relational" became synonymous with "good"; and any other kind of data storage mechanism was prohibited.

I was the lead developer in a startup in those days. Our product measured the quality of T1 communications lines. Our data model was relatively simple, and we kept the data in flat files. It worked fine.

But our marketing guy kept on telling us that we had to have a relational database. He said that customers would demand it. I found that to be a strange claim since we hadn't sold even one system at that time, and no customer had ever mentioned our data storage technology. But the marketing guy was adamant. We just had to have a relational database. Flat files were prohibited.

As the lead developer, responsible for the quality of the software, my view of a relational database was that it would be a big, stogy, slow, expensive pain in the rear. We didn't have complex queries. We didn't need massive reporting capabilities. We certainly didn't need a process with a multi-megabyte footprint sitting in memory and burning cycles. (Remember, this was the '80s). So I fought against this idea with everything I had; because it was the wrong technical solution.

This was not a politically astute move for me. Over a period of several months, a hardware engineer who managed to write a few lines of code, was moved into the software group. He was gradually given more and more responsibility, and was eventually named my co-manager. He and I would "share" the responsibility for leading the software team.

Uh huh. Sure. Right. A hardware guy with no real software experience was going to "help" me lead the team. And what do you think his first issue was? Why it was to get a relational database into our system!

I left a month later and started my consulting career. It was best career move I have ever made. The company I left no longer exists. I don't think they ever made a dime.

I watched the relational database market grow during the '90s. I watched as all other data storage technologies, like the object databases, and the B-tree databases dwindled and died; like the beer companies in the 20s. By the end of the '90s, only the giants were left.

Those giants were marketing up a storm. They were gods. They were rulers. During the dot com bubble, one of them actually had the audacity to buy television ads that claimed that their product was "the power that drove the internet". That reminded me of a beer slogan from the '70s "Ya gotta grab for all the gusto in life ya can." Oh brother.

During this time I watched in horror as team after team put the database at the center of their system. They had been convinced by the endless marketing hype that the data model was the most important aspect of the architecture, and that the database was the heart and soul of the design.

I witnessed the rise of a new job function. The DBA! Mere programmers could not be entrusted with the data — so the marketing hype told us. The data is too precious, too fragile, too easily corrupted by those undisciplined louts. We need *special* people

to manage the data. People trained by the database companies. People who would safeguard and promulgate the giant database companies' marketing message: that the database belongs in the center. The center of the system, the enterprise, the world, the very universe. MUAHAHAHAHAHAHA!

I watched as SQL slipped through every crack and crevice in the system. I ran screaming from systems in which SQL had leaked into the UI. I railed endlessly against the practice of moving all business rules into stored procedures. I quailed and quaked and ranted and raved as I read through entire mail-merge programs written in SQL.

I hammered and hammered as I saw tables and rows permeating the source code of system after system. I hammered out danger. I hammered out a warning. I hammered out that the schema had become "The Blob", consuming everything in sight. But I knew all my hammering was just slinging pebbles at a behemoth.

And then, in the first decade of the 21st century, the prohibition was lifted, and the NOSQL movement was born. I considered it a kind of miracle, a light shining forth in the wilderness. Finally, someone realized that there might just be some systems in the world that did not require a big, fat, horky, slow, expensive, bodily effluent, memory hog of a relational database!

I watched in glee as I saw BigTable, Mongo, CouchDB, and all the other cute little data storage systems begin to spring up; like little micro-breweries in the '80s. The beer was back! And it was starting to taste good.

But then I noticed something. Some of the systems using these nice, simple, tasty, non-relational databases were being designed *around those databases*. The database, wrapped in shiny new frameworks, was still sitting at the center of the design! That poisonous old relational marketing hype was still echoing through the minds of the designers. *They were still making the fatal mistake.*

"Stop!" I yelled. " Stop! You don't understand. You don't understand." But the momentum was too great. An enormous wave of frameworks rose up and smashed down on our industry, washing over the land. Those frameworks wrapped up the databases and fought to grab and hold the center of our applications. They claimed to master and tame the databases. They even claimed to be able to turn a relational database into a NoSQL database. And the frameworks cried out with a great voice heard all over the land: "Depend on me, and I'll set you free!"

The name of this article is “No DB”. Perhaps after that rant you are getting an inkling of why I named it that.

The center of your application is not the database. Nor is it one or more of the frameworks you may be using. *The center of your application are the use cases of your application.*

It makes me crazy when I hear a software developer describe his system as a “Tomcat system using Spring and Hibernate using Oracle”. The very wording puts the frameworks and the database at the center.

What do you think the architecture of that system would look like? Do you think you’d find the use cases at the center of the design? Or would you find the source code arranged to fit nicely into the pattern of the frameworks? Would you find business objects that looked suspiciously like database rows? Would the schema and the frameworks pollute everything?

Here’s what an application should look like. The use cases should be the highest level and most visible architectural entities. The use cases are at the center. Always! Databases and frameworks are details! You don’t have to decide upon them up front. You can push them off until later, once you’ve got all the use cases and business rules figured out, written, *and tested*.

What is the best time to determine your data model? When you know what the data entities are, how they are related, and how they are used. When do you know that? When you’ve gotten all the use cases and business rules written *and tested*. By that time you will have identified all the queries, all the relationships, all the data elements, and you’ll be able to construct a data model that fits nicely into a database.

Does this change if you are using a NoSql database? Of course not! You still focus on getting the use cases working and tested before you even think about the database; no matter what kind of database it ends up being.

If you get the database involved early, then it will warp your design. It’ll fight to gain control of the center, and once there it will hold onto the center like a scruffy terrier. You have to work hard to keep the database out of the center of your systems. You have to continuously say “No” to the temptation to get the database working early.

We are heading into an interesting time. A time when the prohibition against different data storage mechanisms has been lifted, and we are free to experiment with many novel new approaches. But as we play with our CouchDBs and our Mongos and BigTables, remember this: *The database is just a detail that you don't need to figure out right away.*

Robert Martin (Uncle Bob) is a Master Craftsman. He's an award-winning author, renowned speaker, and has been an über software geek since 1970.
