Flexible Schemas Are The Mindkiller

Published on March 1, 2024

We need to talk about the kind of person that loves "flexible" schemas, and why I am unable to disclose what I will do the next time I see one without going on a watch list.

I.

A few years ago, I worked at a company that had received $1M from FAANG (that's Facebook et. al. for the one art historian who reads this blog) to produce an AI and data classification tool. For some of you super slick operators out there in your diamond-clad superstructures, where Kubernetes is a real thing, the phrase "web-scale" doesn't make you assume your interlocutor is a moron, and you spend your days rightfully leering down at the little ants scurrying around from on high, this might seem like a very small amount of money. For this organization, $1M was the *largest* sum of money they had ever received - it immediately went on everyone's CV. Of course, only like three people were working on it, but everyone took credit nonetheless.

Without going into details, this was also the only AI product I've seen at a company I've actually worked at with clear, unambiguous goals that were almost certainly achievable due to the volume of valuable data they had. And to top it off, it had *actual positive societal outcomes*. I was one of the two data scientists, joining the project at a late stage because it had been "lagging a bit". The experienced engineers are already upset. I believe it was in *The Mythical Man Month* that I read a project should "allow no small slippages" - something is either going to be on time or *considerably* delayed, although evidently we still haven't learned from a book written in 1975.

However, the other data scientist was a very close friend, and now co-director at our own company, so I knew that things would be in good shape on the AI end of things. They just needed a hand getting some projects across the line, maybe someone to pair program with. The models were basically already working by the time I joined, so all that was left was to check in on the last member of the team.

Fucking Derek.

II. That Absolute Son Of A Bitch

There were *two* things that needed to be done. The first was the AI stuff (we used BERT encoding and did some deep learning for those interested, but it could have been a random forest, like almost everything). Donezo.

The *second* thing was very simple. The organization had many staff dedicated to data labeling using a series of Access databases. Put simply, it's a database that non-technical staff can use like a spreadsheet for data entry, without all the chaos of untrained people creating your organization's data models for the rest of eternity. Despite the good things spreadsheets enable, please take a moment to reflect on *whether they are actually worth the tradeoff* in light of that sentence.

The organization wanted to replace that Access interface with a simple web application so that some functionality could be added, and new data could be pre-labelled by our machine learning work. To do so, they hired Derek, so named because I evidently decided that Derek is our default name for someone that hurts me. Sorry, Dereks.

Derek had *one simple job*. Make a *really easy* CRUD app.

Create a simple authentication system. Replicate seven CRUD pages. Create a SQL Server instance to store the data instead of Access.

He had been working on it for eight months before I joined, and was scheduled to leave three days after I joined. However, the application was "basically finished" according to management, and Derek had received glowing praise for his extreme efficiency and quiet work ethic. He had also come highly recommended. On the very first day, he is asked to make an adjustment to the application, and he does so in five minutes. He very proudly says this is because the application is "very flexible". **I could have saved myself so much pain if I had run at this second.**

However, since my friend was quite inexperienced at running engineering projects (and I was too, but had some natural aptitude I guess), I decided to make sure the handover was graceful. I asked Derek to share his repository with us.

The rest of this story is one train crashing in slow motion, and it has not actually stopped crashing four years later.

III. It All Goes Wrong

None of the code was version controlled. He had been working for *eight months* and there was not a single commit in sight. It was all just a gigantic pile of files on his computer, and it was all undocumented and uncommented.

I meander away, stunned. I return five minutes later and ask him to just put the whole dump of files in a repository and write up a quick word document explaining how to run the application.

Derek does as asked. The documentation is admittedly adequate. That is the last positive thing that will be said of Derek. He departs soon thereafter.

IV.

It is a month later and... the app doesn't work. It takes us a while to figure out how to launch it, and when we *do* launch it, we realize that the core functionality simply doesn't work. Derek had put together a series of pages that *looked* like they worked to a manager that only investigated for thirty seconds over his shoulder, but they didn't function as intended. Some minor features were missing. For example, it had some small flaws such as *never saving any user input*.

To complicate things further, the whole server was written in C# so bad that I was put off the language for two years until being reassured by competent people that .NET is actually okay, and the frontend was done with Angular so ugly that I *still* think it must suck. It was, of course, a single-page application for no fucking reason.

Well, okay, let's see if we can salvage this. My friend and I start unspooling the mess. The records are being pulled from, uh, *one table*? Wait, what? We have hundreds of columns across those seven pages. The whole database is one table? The whole database is one table. The whole database is *one table*?

What did he *do*?

From Bill Karwin's extremely good book, *SQL Antipatterns*, which I unfortunately only read a year later:

> If you hear phrases like the following spoken by your project team, it’s a clue that someone is employing the EAV antipattern:
>
> • “This database is totally extensible without metadata changes. You can define new attributes at runtime.”

> Relational databases don’t support that degree of flexibility. When someone claims to have designed an arbitrarily extensible database, they’re probably using the EAV design.

He had stored *everything* as key-value pairs. So instead of:

User	Age	Profession
Ludic	29	Tortured Soul

We had:

User ID	Key	Value
1	Name	Ludic
1	Age	29
1	Profession	Tortured Soul

Each individual case to be labelled had 350 rows of data now, in one *super long* table. Remember how I said this was actually a reasonable AI project? Yeah, we had a *lot* of data, now multiplied by 350 and a fucking nightmare to query.

I opt to proceed by redoing the entire database. It takes *a while*. Then we rework the Angular app (also *very* painful) to work with the new (correct) data model. The decision to just do the right thing here instead of limping on forever gave us *huge* speed benefits later on, and I think it's one of my proudest professional accomplishments even though it sounds simple. It was boring, painful, and no one other than my friend would ever appreciate it. I *knew that* going in, and I still did it.

Incidentally, I was allowed to do so only because management had been so negligent in allowing things to get to this state that they also didn't notice I was getting them out of this state, and thus couldn't insert their asinine opinions on how good practice is actually too slow. We've got to *ship*.

V.

Oh, you thought this was over? No, no, you see, the flexible schema is simply a portent of things to come. It is the blood moon hanging low in the sky, a malevolent eye bleeding crimson despair across the landscape. It is the sickly glint of a ravenous knife in a dark alleyway. Wielded by the uninitiated, it is a sign that you must flee, flee as fast as your legs can carry you, and I was too foolish to do so.

Each morning, I'm one of the few people that takes the time to get to know the data entry team, and it is becoming increasingly clear that they are treated by management as not-quite-real-people. Useful, to be treated with cordiality, but certainly not to be invited out for morning coffee. The *core* functionality is brought into a state where I know we can deliver on time, and I cheerfully mention this right before lunch.

"Oh, amazing, can we see it? We've been waiting years."

... *Years*? We only got the $1M about nine months ago. Development hadn't started. How long has management been promising this app to these people? I didn't really have time to look for these signs of dysfunction when I joined because of my visa status *and* not knowing what to look for those days.

Wait. No one *had shown the staff the application in ten months*?

Yeah, it turns out they just showed Derek the Access database and said "Could you turn this into an app?". And then they didn't talk to him for eight months or do anything remotely close to requirements gathering. Just let one guy that didn't know anything about the company's operations just start designing the software that would underpin the entirety of their operations. Cool. Cool, cool, cool.

I show the staff the application, and I think they're more thrilled that someone has actually included them than they are at the application. We make some tweaks because it is immediately apparently that it still has major flaws. This takes a bunch of time too, and I am *tired*. This is all happening amidst the middle of Covid, so I am periodically being forced to spend weeks locked in my apartment.

VI.

It is around 11 AM, about a week later.

I am working on a schema migration. I look at one of the Access databases that Derek had been populating his local SQL Server instance with.

Very abruptly, my blood runs ice-cold, something I had only experienced *once* before when I endured the rite-of-passage of my first production mistake. I have no idea how this hadn't occurred to me until this second, seven months after Derek left, but suddenly the right two neurons fire. My blood is running cold *now* just thinking about it.

He had copied one of the Access databases onto his local machine. An Access database full of *extremely sensitive* medical data. Like "I have never heard of a leaked dataset that would be this damaging to vulnerable people" levels of sensitive. And I have a copy. Which means *Derek put it on GitHub*.

There has been a voice playing in my head since that moment. It has become very quiet, and I had almost forgotten about it, but I can hear it now. It sounds like this:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.

Just take a little bit to soak in the horror for a bit. I'll wait.

...

...

...

Okay, we're done panicking together. Let's move on.

The repository is, thankfully, set to private. Which of course, doesn't change the fact the fact a fuckton of absolutely radioactive data had just been uploaded to a third-party's servers outside of the country.

We handle it very judiciously. I can't even remember how it was raised or what the decision was, but this was out of my hands as a lowly brown person on a very tenuous visa. It would be wrong to say it blew over because management finessed it such that there was nary a whisper of any wrongdoing. I am pretty sure it all worked out fine, truth be told - at this point I suspect all the backups containing the breach have expired, and we hit the actual file with the BFG and then ran all sorts of things on our hard drives to ensure that the data wouldn't be recoverable even by specialists. Enterprise infosec isn't a fake circus moon though, am I right fellow executives?

I also *just realized* that he might have been working on a personal laptop. That we wouldn't have scrubbed clean.

I need to take another moment to calm down.

VII.

It is a week later. I am deep in horrific depths of some helper function Derek wrote to produce some of SQL Server tables. There is a comprehensive docstring at the top of the file, which is very unlike Derek. Maybe he isn't so bad.

> COPYRIGHT MEDTECH, 2015.
> Author: Derek

That... isn't where I work.

I stare at the screen for a few minutes, pondering what to do.

I close the file and move on. Just... just... yeah.

VIII.

The next day, management announces that remote work is cancelled, and we need to be in three days a week because it will aid collaboration. Our addle-brained senior manager, who has no idea what's happening but also put that $1M on their CV, keeps forgetting whether they're usually honest with me or not. Each day, they flip a coin and decide whether they're going to say:

> I am so excited to come back to the office! It'll be great to see some people around again, great for productivity.

And:

> I hate this, but the executives are making us do it. God, I would prefer to be spending time with my grandchildren.

But the air conditioning in the office doesn't work very well, and I have suddenly remembered that I enjoy not being in a dimly lit room at 27 degrees Celsius (80.6 degrees Fahrenheit for you whacky Americans) for eight hours a day surrounded by lizard people. By sheer luck, my permanent residency is approved after years of struggle. I quit.

The most important person in the organization calls me to yell because they made me promise to stay in order to not get deported a while ago. Sometimes I think about their impotent fury paired with the flaccid repetition of "I just thought you were a man of your word but I can see I was mistaken" and smile.

In retrospect, I would be livid that I did *so much* to help them. But firstly, I was well-paid. Secondly, the other engineer quits too within two months. One of my current clients, doing lifesaving work that my team agreed to help pro-bono (and I'm so proud of them for it), is in the periphery of that project. They confirmed recently that it still hasn't launched despite there being only a year of work (*extreme* upper bound estimate) left to launch at the time I left for two competent engineers.

Perish, nerds.

IX.

> “Bad programmers worry about the code. Good programmers worry about data structures and their relationships.” - Linus Torvalds

What about the programmers that do neither?

I do not know what it *is* about flexible schemas, but there is a certain type of person that probably has all the mental horsepower required to be a *phenomenal* engineer that simply gets stuck on their "elegance" and then never progresses any further. The fact that Derek had the working memory and patience to whip this grotesque Angular/.NET/schemaless abomination into something that even ran at all indicates he has a level of pain tolerance and raw brainstuff that I cannot even approach. I *can't handle* things that badly designed, and it isn't just aesthetic sense or design intuition. The hardware in my skull isn't up to spec.

But I just *keep meeting that person*. The first sign I'm dealing with an Idiot-Pattern clone is *always* "I've found a way to not worry about our data model, ha ha", and always ends with "I have single-handedly destroyed the infrastructure of this company". Also, you can never fire them because there *is* a schema, it lives in their incomprehensible code and in their brains. The worst part is that they're *always so proud* of the flexibility, as if they're the first fucking genius to realize that if you discard things like, I dunno, database types and queryability and any sort of validation, then you can code "without interruption" whatever that means. Then they exhibit this lack of judgement in every other area of their professional lives.

I *do not* understand how they are both smarter than me in many respects, and then *still don't understand how stupid this all is*.

X.

It is years later. I am brought onto a platform that is advertised as state-of-the-art. Every competitor in the state is copying our implementation. On my first day, they tell me that all ETL metadata and logging is stored in DynamoDB because it is more flexible.

This was the first clue that they were idling their computers to the tune of half a million dollars.

Ahem.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.

PS: You can stop reading here, the post is over. The rest of this is about setting up Liberapay and Patreon to receive support for writing/podcasting/whatever.

The preferred option is Liberapay because it is open-source and aligned with the principles of this community. However Patreon is also available in case you already have an account and just want to click the buttons fast.

The reasons for this are written in this post. However, you will probably get my writing forever, for free. The "probably" is not because I'd start charging, but because life might get in the way at times, and there might be breaks in my output. And the currently-being-recorded podcast will also probably be free forever, though this might actually have some paywalled episodes one day if it is time-consuming enough - editing audio doesn't sound as fun as writing. But of course, even then I'm promising right now, with God as my witness, that I'd hand it out for free to anyone that can't justify the expense because I have something called *ethics/bad business sense*.

Currently, you will receive no benefits other than showing appreciation, covering the cost of extending my weekly music classes from 30 minutes to 45 minutes, and eventually paying for hosting a cheap server or two that I'll use to deploy some stuff to help coordinate our community.

Both systems are configured to show how much I'm getting even though this is demonstrably *terrible* for your income, because I am evidently hellbent on only accepting money from *very insistent* people.

I am only going to add this footnote to a post once every six months (this must be done or no new readers will ever know the links are there). And then I won't have to mar too many posts with this stuff.

Thanks for reading!