The next frontier in gender rights is inside databases.

Meredith Broussard Oct 23, 20193:12 PM

When Binary Code Won't Accommodate Nonbinary People

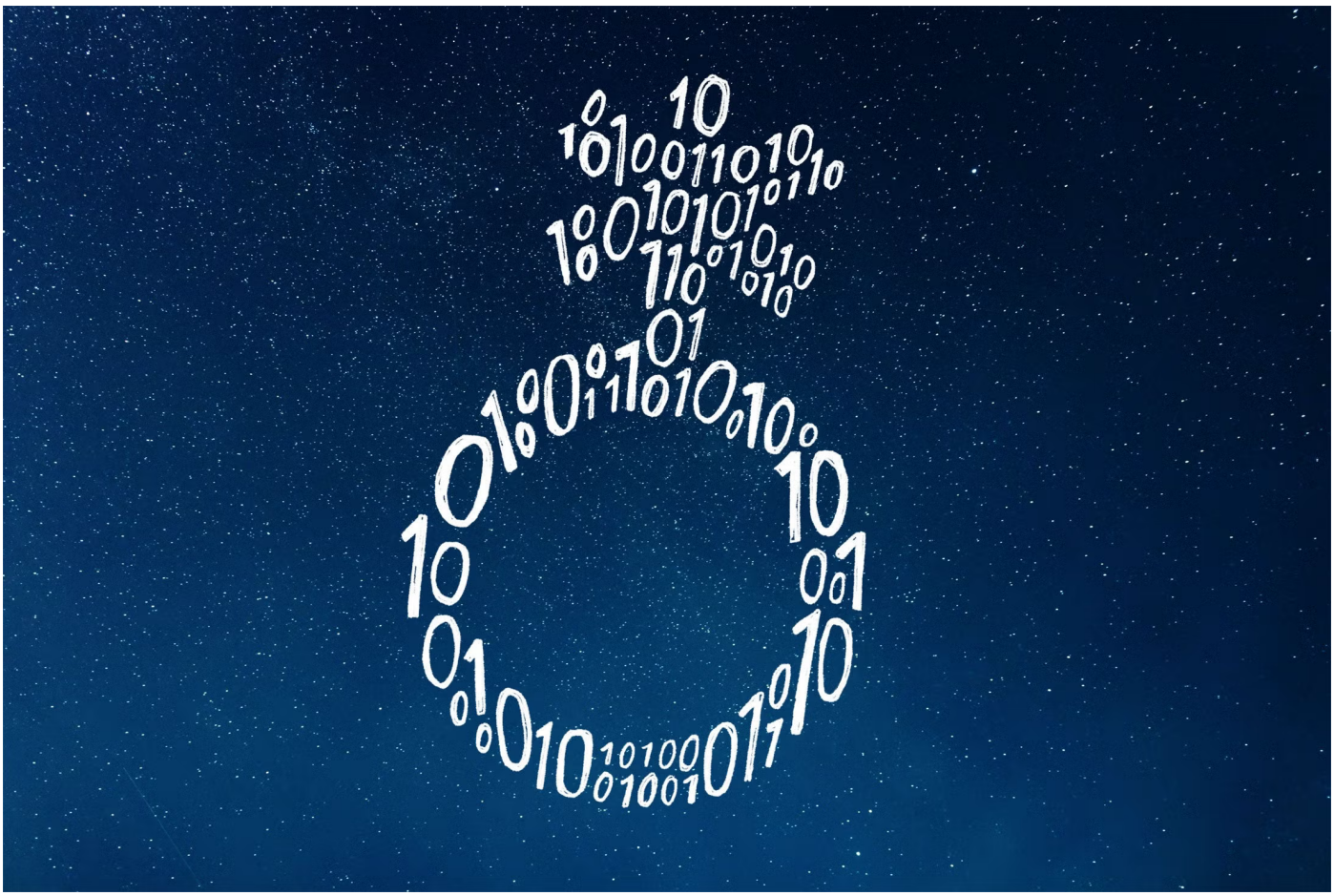

Photo illustration by Lisa Larson-Walker. Photo by Allexxandar/Getty Images.

College student Manahil Bandukwala recently tweeted a complaint about her computer science class. In a gesture of solidarity with her trans and nonbinary classmates, she wrote that a professor had said that "programs can only have two genders and you can't change your gender and how

people changing their gender broke the university's system…as though trans and enby folx are an inconvenience to code." But the real problem, she said, was "shitty code erasing [people's] identities."

Back in the 1950s, when modern computer systems were first designed, gender was generally considered fixed. If you filled out a paper form, it asked for your name and offered you two choices for gender: male or female. You could pick one.

This was how computer database design was taught through the 1990s, when I learned programming. "Back then, nobody imagined that gender would need to be an editable field," one friend said recently. Today, we have a more comprehensive understanding of gender, and an increasing number of companies are allowing users to self-identify in databases as nonbinary, transgender, genderqueer, and other terms that encompass a range of LGBTQIA+ identities. However, artifacts and idiosyncrasies inside computational systems serve as barriers to implementing truly inclusive design. Most of these problems come from the way that 1950s U.S. and U.K. social perspectives informed how computer schemas were created. This is one of the many situations where a battle over social norms is being waged through code.

Whose values are encoded in the system?

Bandukwala is right: Many computer scientists and engineers are personally and professionally committed to the gender binary and cisgender, heterosexual norms. Over and over again, we have seen technical problems arise because the people designing computer systems were committed to replicating a rigid, retrograde status quo. When same-sex marriage was legalized in the U.S., it prompted a database redesign called Y2gay. Most

databases were set up to only allow marriages between men and women; changing the law required changing those databases to comply. Facebook, which began as a kind of "Hot or Not?" for male undergraduates to rate women, requires users to commit to a gender identity at sign-up: female, male, or "Custom." In 2014, Facebook became one of the first social media companies to allow users to change their names and gender identity, with more than 50 options. Although its software seems to allow users to seem to self-identify, the way the system actually stores the data is that each user is recorded (and sold to advertisers) as male, female, or null.

The reason for this has to do with both hegemonic heteronormativity and math. Everything you do on a computer is secretly math, and that's the trouble. The messiness of the "real" world and people's shifting identities are rarely consistent with the sleek empiricism required to effectively do the math that is under the hood in computers. This is most obvious when it comes to the gender binary and binary representation in computer systems.

You know the gender binary: the idea that there are two genders, male or female. Binary code is also the system that powers computers. In a binary numeral system, there are only two numbers: 0 and 1. The numbers 0–4 look like this in binary:

0	0
1	1
2	10
3	11
4	100

When binary information about the world is stored in a computer, we call it

data. Data is stored inside a database. In a database, every piece of data has a type, and usually the rules for that type are very strict. In the very simplest form, we can think of data as being of three types: letters, numbers, or binary (0 or 1) values. A binary value is often referred to as a Boolean, named after a 19th-century guy named Charles Boole who invented a system of logic that only uses 1s and 0s. If you want to use data in a computer program, you feed that data to a thing in the program called a variable. Variables also have types, and those types are strictly governed by the rules of a specific programming language. Variable types are slightly different in Python, a more modern programming language, than they are in C, which was developed in the 1980s. Unlike human languages, programming languages have very strict grammar and vocabulary; all programming languages have the same essential forms, meaning that they all on some level translate keyboard strokes, mouse movements, variables, data, etc. into binary. This is why a computer can't work without power. Despite all the magical thinking about what computers do, ultimately a computer is a machine that merely uses electricity for calculation.

So: In order to store data in our electrically powered poison rocks (as futurist Ingrid Burrington recently referred to computers), we have to declare variables of a certain type inside a database. Speaking loosely, the types are string (meaning text, as in a string of letters), number, or binary (aka Boolean). Boolean variables are used when a value is true or false, and it's represented as 1 or 0. 1 is true, 0 is false. That looks something like this:

Firstname [string]

Lastname [string]

Gender (M/F) [Boolean]

Address 1 [string]

Address 2 [string]

Zip [number]

In the sample database record above, we'd have to make certain decisions about each field—decisions that can become very loaded. What type of data goes into each field? How large does each field need to be to hold the intended data? Who can enter the data? Who can change the data? Under what circumstances can the data be changed? Which fields can be edited, and which are fixed? Usually someone makes a recommendation, then there are meetings in which people go over the data fields and talk about scenarios like, "What happens when someone gets married or divorced, and changes their name?" This is not about math, but about human social values being superimposed on a mathematical system. The question becomes: Whose values are encoded in the system?

Even something as seemingly small as choosing free text entry versus a dropdown has implications. A letter is a bigger number, which occupies more bits and thus takes up more memory space. Today, it's easy to ignore memory concerns, but until the late '90s, computer memory was expensive. I was taught to write programs that were as concise as possible, then to refactor those programs down so they took up as little memory space as possible. There's something satisfying about it, writing code that is small and runs fast. Using a Boolean variable is extremely efficient. 0 or 1 takes up less space than 01001101 (M) or 01000110 (F).

If you are designing code for maximum speed and efficiency using a minimum of memory space, you try to give users as few opportunities as possible to screw up the program with bad data entry. A Boolean for gender, rather than a free text entry field, gives you an incremental gain in efficiency. It also conforms to a certain normative aesthetic known as "elegant code."

That aesthetic, however, dates to the very earliest era of computing. It's not inclusive. It is specifically exclusionary to someone like Zemí Yukiyú Atabey, an NYU graduate student who identifies as genderqueer and nonbinary. Atabey's pronouns are ze ("Where is ze?")/zem ("I don't have the tickets. I gave them to zem."). "As a nonbinary person, there is no option most of the time," ze says of entering personal information in databases. "There's only male or female, which doesn't fit my reality or identity." Microsoft Word, the program I used to compose this story, marked all of Atabey's pronouns with the red squiggly underline. Meaning: The people at Microsoft who wrote Word do not recognize Atabey's pronouns as acceptable English words, even though the genderqueer community has been suggesting the use of *ze* and *hir* as pronouns for at least 20 years.

"While issues of identity, data, and information systems seem to be—on one level, at least—an interesting conceptual or philosophical problem to ponder, they also expose the urgency of recognizing the very real and lived challenges these tensions and the rapid rise and adoption of data-intensive technologies and platforms generate for already vulnerable trans and queer populations," writes University of Washington professor Anna Lauren Hoffmann in "Data, Technology, and Gender: Thinking About (and From) Trans Lives." That trans and gender nonconforming people are excluded from or subjugated to information systems is a phenomenon she labels data violence, or "Harm inflicted on trans and gender nonconforming people not only by government-run systems, but also the information systems that permeate our everyday social lives."

NYU, my employer, is among the most progressive universities when it comes to gender identity. Students can change their gender identity in Albert, the student information system.

In the Albert documentation, a distinction is made between legal sex and

gender identity, which is also the recommended best practice in electronic health systems.

Making this possible was a complex matter. Universities' student information systems are core—everything feeds off of them. Dozens, even hundreds, of other systems and programs feed data back and forth to Albert every hour of every day. But most of these systems were set up in the 1960s. Remember, variables are of strict types. Let's say that you have an old system where you have a field name Sex, of type Boolean. If you then change the field name to LegalSex of type string, and add another field GenderIdentity of type string, you break the system because the other programs are looking explicitly for Sex, which is a Boolean. You can't pass a Boolean to a function that expects a string, and vice versa.

We may think of computers as nimble and agile, but in reality, changing legacy systems is complicated and expensive. As NYU demonstrated, though, it's absolutely possible to change any system that relies on legacy design. It's a matter of will and funding—and in computer science, those can be in short supply when it comes to recognizing the not-so-binary world we live in.

Future Tense is a partnership of Slate, New America, and Arizona State University that examines emerging technologies, public policy, and society.

Gender
History of Code