What's a relational database?

This table did have relations with that table

Justin
May 11, 2020

The TL;DR

A relational database is sort of like Excel, but for developers: it's how applications store and analyze data.

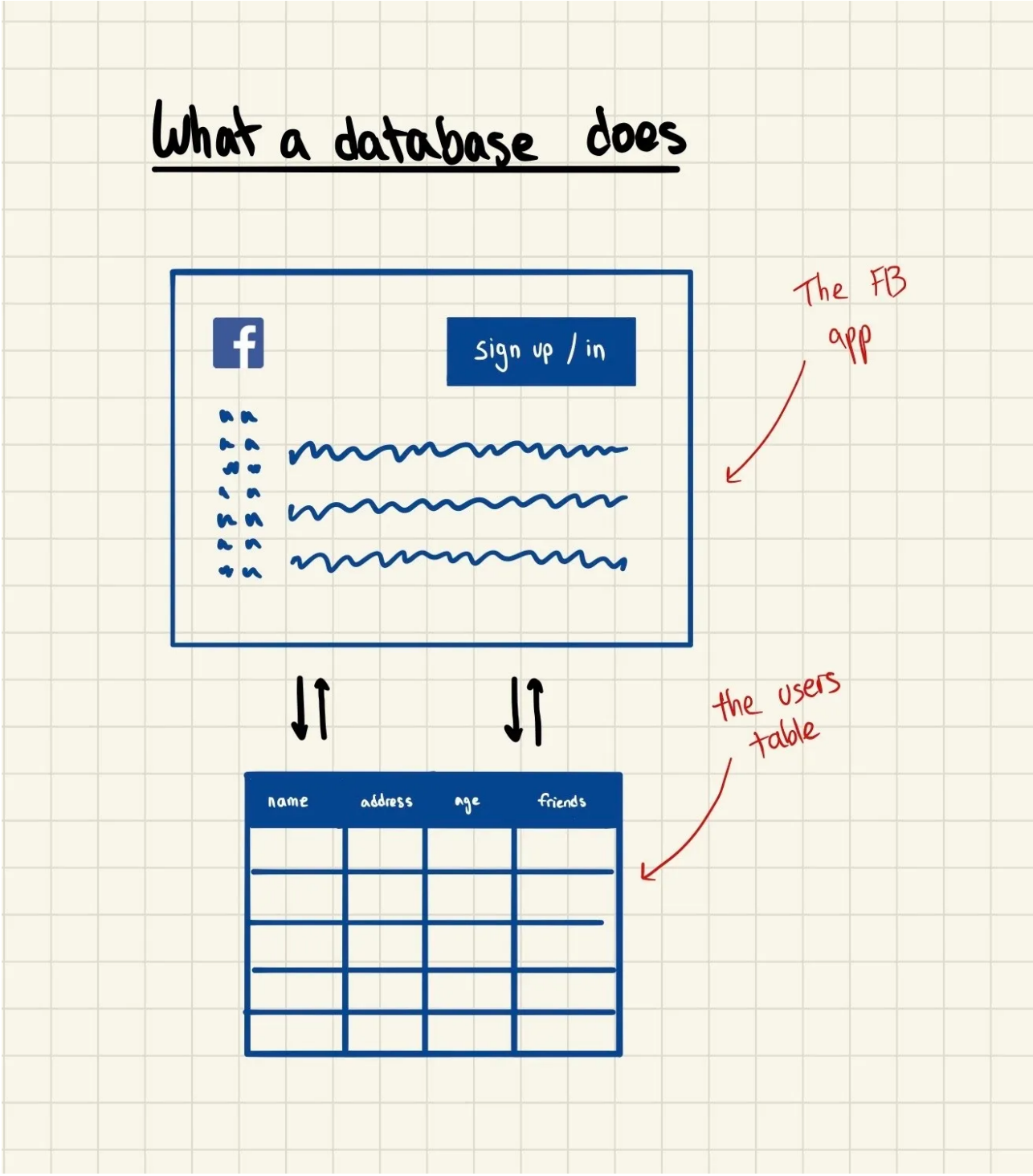
- Relational databases like MySQL and PostgreSQL are the **default software for storing data**
- Relational DBs are **all about schemas**: rules for how data is stored and organized
- Inserting data into relational DBs follows the **ACID specification**
- **SQL** is a type of programming language that lets you **pull and analyze** the data in your relational DB

Relational DBs power at least part of almost every single application that you use in your day to day. So read this.

What's a relational database?

→ **First: what's a database in general**

A database is just a place to store data; even Excel is technically a database. But when developers talk about databases, they're usually referring to *specialized software* that's used to store data from *applications*. When you sign up for Facebook, a new entry gets added to Facebook's users database with a bunch of information about you; every time you sign in after that, that data gets pulled and used (this is part of what a backend is).



There are thousands of different databases out there, all optimized for different types of applications, some free and some paid. Some popular ones you might have heard of: MySQL, PostgreSQL, and MongoDB.

> **🚨 Confusion Alert 🚨**
>
> People use “database” to refer to a lot of different things: most of the time it’s our topic (programmatically accessed data stores), but you might also hear

> about “my database of top CEOs” or something like that, which is probably a spreadsheet. Just listen for the context and you should be fine.
>
> 🚨 **Confusion Alert** 🚨

→ **Second: what’s a relational database**

Relational databases are the most popular kind of DBs, and they’re pretty simple: they add **rigidity and rules** around how data is stored. To understand what that means, let’s imagine we’re moving across the country (*yes* I’m still crying, *no* I don’t want to talk about it). You’ve got a ton of stuff in your house: clothes, furniture, random keychains that you can’t let go of, and kitchenware, among other things. How do you store / pack all of that?

- You could throw everything in random boxes, like back in college

or

- You could organize things by size or category, and pack and label specific boxes

There’s a **clear tradeoff** here between speed and efficiency. Packing properly takes a lot more time, but allows you to make better use of space and more easily find your stuff later. It turns out that databases have the exactly same tradeoff, and relational DBs are like packing properly instead of throwing stuff in.

In relational databases, you specify everything about your data up front:

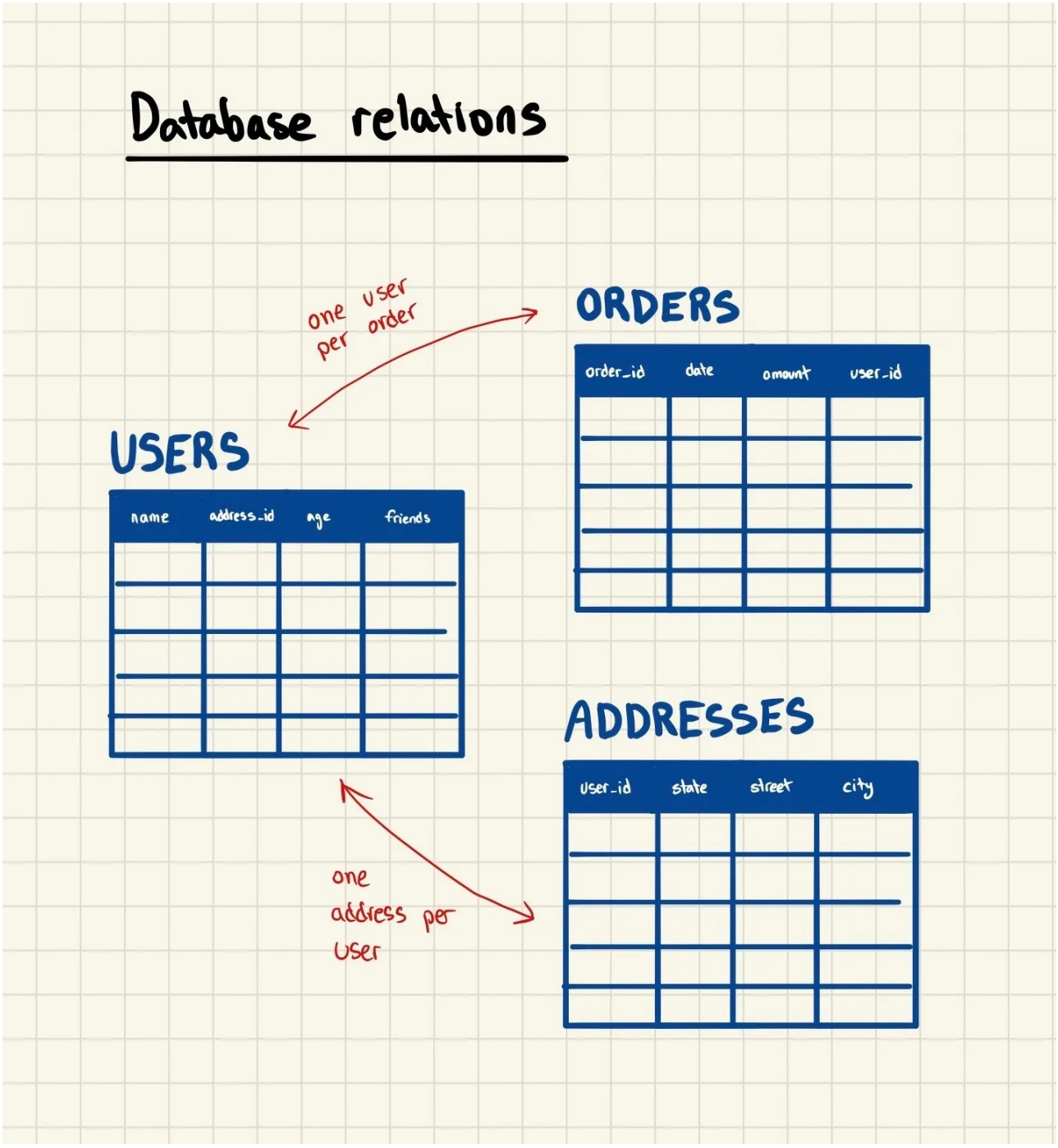
- What type it is (text, number, yes/no, etc.)
- How it’s organized (different tables, how those tables are related to each other, etc.)
- Names for everything (columns)

All of this together is called a **schema**, and it’s the basis of relational DBs. But why are they called relational?

Relations, take 2

A relational database is called what it is because of the *relationships it models* between your data.

In a relational DB, data is organized into **tables with rows and columns**: you might have a table for data about your app's users, as well as a table for your app's orders. Those tables are related, because each order has a user that made that order. Big applications can have **hundreds or thousands of those tables**, and you need to define the relations between them for it to work well.



In this diagram (and, I'm guessing, in Amazon's internal systems) there's a users table and an orders table: the *relation* between those tables is that each order has one user who created it.

There's a good reason why things are designed this way instead of throwing all of your data into one giant table: it makes inserting and pulling data *way* easier and more efficient. In the early days of the internet, when this technology was just starting out, it was *incredibly* expensive to store and retrieve data. So people spent a ton of time on optimizing it: making sure the same data is never stored twice (this is called **normalization**), finding the most efficient way to store different data types, and all of these low level things that are beyond my pay grade.

Transactions and SQL

Just in case there weren't enough acronyms yet, I gotchu.

→ ACID transactions

Modern software applications need to be *fast* and make few or no mistakes. Facebook can't mess up your login, or mistake you for another user: that's why relational DBs have special features around **transactions**. A transaction is just an interaction with the database, either **inserting data** into it or **querying and pulling** data from it. Transactions can go wrong *all the time* for a ton of different reasons, but there's *a body of standards called ACID* (broooo) that helps avoid them.

The specifics of ACID aren't too important. Just keep in mind that if you see a phrase like "ACID compliant transactions" it's referring to these standards.

→ SQL

Pulling data from a database is called querying, and there's a series of special programming languages that let you do that called Structured Query Language (SQL). There are all different types and flavors of SQL (depending on which relational DB you're using), but they're all pretty similar. A typical SQL statement might look like this:

```
SELECT
    first_name,
    last_name,
    address,
    last_order_date
FROM users
WHERE signup_date > '2019-01-01'
```

SQL is pretty human readable – even if you don't know it, you can guess what we're doing here – and it's a really useful thing for everyone to know, even if you're not en engineer.

> ☕ **Dev Culture** ☕
>
> Software developers need to know basic SQL for building their apps, but the *biggest* SQL experts are usually Data Scientists and Analysts: their entire days (I was one!) are spent querying and manipulating data from databases. For that and a few other reasons, some developers treat SQL as a second-tier, not "real" programming language.
>
> ☕ **Dev Culture** ☕

Terms and concepts covered

```
Database
```

```
Schema
```

```
Relations
```

```
Normalization
```

```
Transaction
```

```
ACID
```

```
SQL
```

Further reading

- As data storage has gotten considerably cheaper, new types of databases that *aren't* relational have started to take off, like MongoDB
- There's a ton of annoying overhead to running databases in production, so major cloud providers like AWS offer expensive ones that simplify things and save developers time

---

2 Comments

Write a comment...

Jeff May 13, 2020 ♥ Liked by Justin

I was about to make some snarky comment that all apps are just the front ends of Excel Spreadsheets but then you had to tease non-relational DB at the end 👀

♡ 1 Reply Gift a subscription Collapse •••

1 reply by Justin

1 more comment...
