What's NoSQL?

What could this esoteric name possibly mean?

Justin

Jun 15, 2020

The TL;DR

NoSQL databases are databases with no rules: you just throw your data in there and worry about it later.

- NoSQL is a class of databases built for **large amounts of data** and fast application speeds
- Unlike relational databases, NoSQL works **without a schema**, or set of rules and definitions
- Most NoSQL databases are built to be **resilient and distributed** as opposed to transactionally sound

NoSQL has *really* taken off over the past few years, and almost 30% of developers are using it in one way or another.

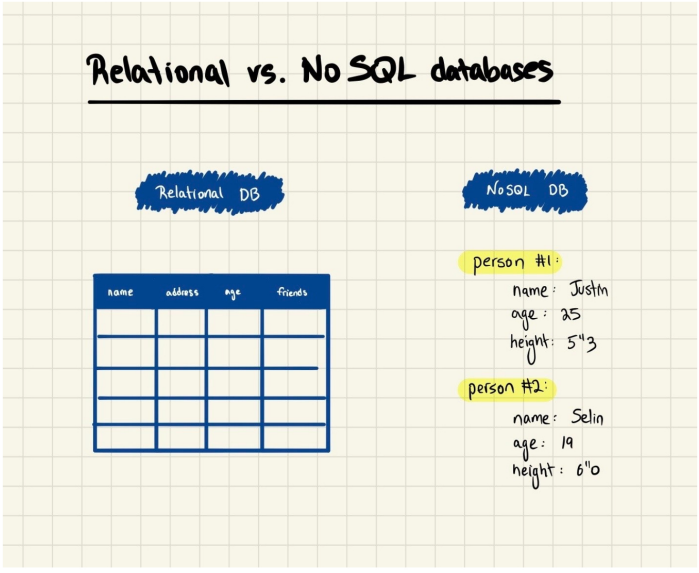
> 🔮 **Dependencies** 🔮
>
> To get the most out of this post, you'll need to understand what a relational database is and how software runs in the cloud.
>
> 🔮 **Dependencies** 🔮

What's NoSQL?

When we covered relational databases, everything was about **schemas**: you set all the rules that your data needs to follow up front, and that yields a lot of important benefits for inserting and querying data down the road. Schemas are great, but like any rules, they can get really annoying sometimes:

- You need to define a ton of stuff up front, which is a lot of work
- Data and applications change all the time, so you need to update your schema constantly (which is really hard)

It's the same tradeoff as throwing all of your stuff into a suitcase vs. packing it meticulously; if you're rushed, or if you've got plenty of room, you might want to skip the folding and organizing. That's exactly what NoSQL is (literally, you don't use SQL to query it) – it's a database without all of the rules. You can insert data through a much more flexible and simple process, and read it out more directly.



There's a reason that NoSQL hasn't gotten particularly popular until recently: it's gotten so cheap to store and access data that we don't need to be super stingy

anymore. Overall, running your database without a schema and normalization is going to use much more space and be less efficient, but that doesn't always matter anymore. If it saves developer time, it might be worth it. That, and NoSQL databases are *much* easier to scale.

> 🙏 **Workplace Example** 🙏
>
> Today, a lot of companies with serious applications running use a combination of relational and NoSQL databases: more than 25% of developers use MongoDB alone. Ask your developers which types of databases they're using and why they chose them.
>
> 🙏 **Workplace Example** 🙏

Types of NoSQL DBs

Ok, so you've packed your stuff completely haphazardly, but you did it fast. Now how do you get your jacket out of there? Even though NoSQL is technically schema-less, there needs to be *some* structure around how you insert and query data, otherwise it would be completely useless. NoSQL DBs approach this in 4 distinct ways

→ Key value stores

A key is the label, and the value is whatever you're storing. In key value stores, you label any data that you want to store (order #1, person named Justin) and then retrieve that data by using the key. Key is kind of a weird word to use here, but hey, I didn't write this stuff.

```
{
name: Justin,
height: 5”3,
weight: “wouldn’t you like to know”
}
```

→ Document DBs

A document structure is just like a key value store, but with *layers*: you can have a value that has keys and other values in it.

```
{
    guitar_players: {
        justin_gage: {
            height: 5"3,
            weight: "wouldn't you like to know"
            },
        jake_mendel: {
            height: 5"7,
            weight: 160
            }
        },
    drums_players: {
        chris_behrens: {
           height: 6"0,
            weight: 165
        }
    }
}
```

→ Wide column stores

These are just like normal rows and columns, but unlike relational DBs, they can change at any time. It's sort of like a two-dimensional key value store, if that helps you understand it better.

→ Graph DBs

When you're storing data about how things are *connected* to each other (like how Facebook tracks who's friends with who), there are special NoSQL databases like Neo4J that are organized like graphs.

> 😇 **Don't sweat the details** 😇
>
> It's OK if you don't understand the exactly differences between these types of data stores. Just keep in mind that NoSQL is a very broad category, and there are specific types of NoSQL DBs for specific use cases.
>
> 😇 **Don't sweat the details** 😇

How NoSQL actually works

Another important feature of NoSQL databases is how they actually run. Most

NoSQL DBs have a few nifty things out of the box that make life easier for developers.

→ Sharding

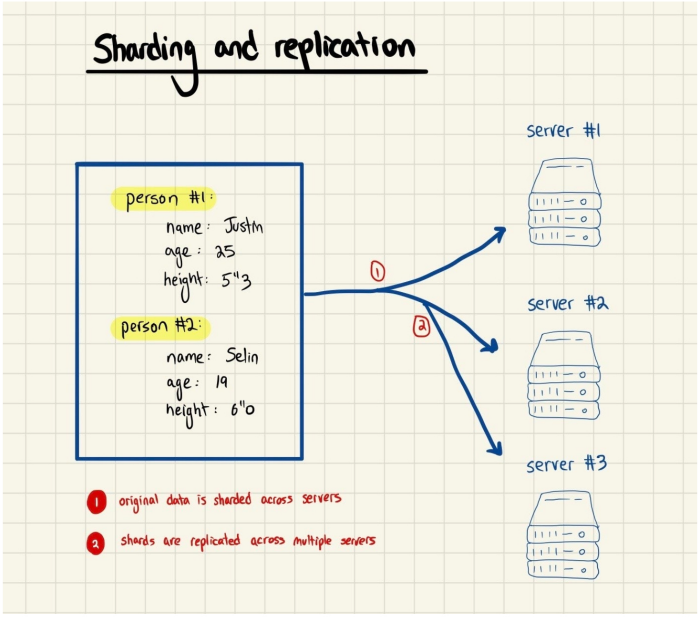
When we covered cloud, we talked about how for most applications, the actual *hardware* running the code in question – in our case, the database – is on some powerful server somewhere far away. Historically, one program (i.e. one database) would run on one machine; but over time, *distributed systems* have gotten more popular. That means that you'll actually run little pieces of your database on multiple machines, and that's what **sharding** is.

Sharding stores different parts of your data store on multiple different servers. The reason you'd *want* to do this is that it can make query performance much quicker than storing everything together, but more on this a different time.

→ Replication

You and your grandma know more than anyone that *a lot of things* can go wrong with computers. When you're storing application critical data on them, that would be pretty catastrophic: that's why developers **replicate their data across multiple servers**. That way in case anything goes down, you never lose your data.

If you're using a relational database, replication isn't very simple: you need to follow a rigid transaction model (remember ACID?) among other things. NoSQL's architecture makes replication much easier, and it ships as a default feature of a lot of popular NoSQL products in the market.



→ Querying

If you don't have SQL (remember, *No* SQL), how do you query data in NoSQL databases? There's no simple answer, and it varies depending on which database you're using. Usually it will be through some kind of programming language (like Javascript) and some built in functions. MongoDB is the most popular NoSQL database on the market, and they have *drivers* (little pieces of software) that you can use in multiple different programming languages to write queries.

Terms and concepts covered

```
NoSQL
Schemaless
Key value stores
Document stores
Wide column stores
Graph databases
Sharding
```

```
Replication
```

Further reading

- Over the past few years, some breakthrough innovations in relational DBs have borne products like Spanner that are giving NoSQL DBs a run for their money
- MongoDB is a NoSQL document store, and it's the most popular NoSQL database out there with >25% adoption among developers

See all ›