Postgres vs. MongoDB for Storing JSON Data – Which Should You Choose?

Eran Levy
November 22, 2017

Have you ever tried to eat pasta with a spoon? It doesn't work so well. With nothing to grip your fettuccine or your penne with, it slides all over the place. A fork, on (in?) the other hand, is obviously a million times better: you can spear and scarf your pasta as fast as you like.

Except, when you get to the bottom of the bowl and want to scoop up the last of your sauce, a fork is useless. Now you need a spoon again – meaning you need two tools for what *should* be a simple task.

That was, however, until some smartypants invented the **spork**. Suddenly there was something out there which could do the job of a spoon and a fork. So why don't we all use sporks for every meal? Why do both forks and spoons still exist at all?

When you get down to it, this is precisely the debate that rages between data scientists when it comes to PostgreSQL vs MongoDB, and the right kind of storage for JSON data.

In the past, the Postgres vs MongoDB debate looked like this: you had Postgres on one side, able to handle SQL (and later NoSQL) data, but not JSON. On the other, you had purpose-built database management systems (DBMS) – like MongoDB, which was designed as a native JSON database.

Today, though, this strict separation has been muddled by the advent of a bunch of in-between options; the data sporks, if you will.

The SQL-rooted database architecture PostgreSQL now offers enhanced JSON storage capabilities. So why would you opt to keep both tools?

The Rise and Rise of JSON and JSONB

Before we get into that, let's remind ourselves what we're dealing with here. What is this JSON data format we're trying so hard to accommodate?

JavaScript Object Notation (JSON) is unstructured, flexible and readable by humans. Basically, you can dump data into the database however it comes, without having to adapt it to any specialized database language (like SQL). You can nest fields in a data record, or add different fields to individual data records as and when you need.

All of this makes JSON an important step towards user-friendly computing. Today, many prefer it to XML, and the JSON data format is used by a number of NoSQL data stores.

JSON does, however, lack indexing – and the JSONB data format was created to tackle this problem. JSONB stores data in a binary format, instead of a simple JSON blob. Data input is bit slower, but processing becomes a lot faster since the data doesn't need to be reparsed.

What is MongoDB? What is PostgreSQL?

Okay, now that we're clear on what we're working with, let's take a look at the differences between these two commonly used databases.

MongoDB is an open source database. It's designed to be agile and scalable, and it uses dynamic schemas so that you can create records without defining the structure first. It also supports hierarchical documentation of data.

PostgreSQL is also open source, but it's a relational database that is much more concerned with standards compliance and extensibility than with giving you freedom over how you store data. It uses both dynamic and static schemas and allows you to use it for relational data and normalized form storage. MongoDB, with its unstructured approach, can't do that.

So… which one should you use to store your JSON / JSONB data?

Deliberate Constraints and Collateral Limitations

First, to be clear, Postgres and MongoDB **both** have functions for JSON and JSONB data storage (although MongoDB calls the latter “BSON”).

There are differences, though:

- MongoDB limits its BSON format to a maximum of 64 bits for representing an integer or floating point number. Postgres’ JSONB format isn’t limited.
- Postgres provides data constraint and validation functions, which help to ensure JSON documents are more meaningful. E.g. It stops you storing alphabetical characters when only numerical values make sense.
- MongoDB offers automatic database sharding, for easy horizontal scaling of JSON data storage; Postgres installation scaling is usually vertical. You can scale Postgres horizontally, but this tends to be trickier or takes third-party help.
- MongoDB also lets you increase your write throughput by deferring writing to disk. You might lose some data that way, but it can be good for users that are less worried about persisting their data.

The big thing, of course, is that Postgres lets you keep your options open. You can choose to route data to a JSON column, allowing you to model it later, or you can put it into an SQL-schema table, all within the same Postgres database.

So, spork option it is then? Well, not so fast, because…

Native JSON Data Stores do not always have the Best Performance

One of the best things about NoSQL database management systems is their performance.

Since they work with simpler data structures than SQL databases, storage and retrieval tends to be faster in NoSQL database systems.

While they may lack the ACID (atomicity, consistency, isolation, and durability) properties you need for financial transactions, etc., they’re great for handling large volumes of unstructured data, at speed.

That said, Postgres gave everyone a shock by beating MongoDB’s performance ratings on EnterpriseDB.com in 2014.

You read that right. Incredibly, in tests based on selecting, loading, and inserting complex document data to the tune of 50 million records, Postgres was around twice as fast at data ingestion, two-and-half times as fast at data selection, and three times as fast at data inserts… all while consuming 25% less disk space.

In fairness, MongoDB 3.0 has since risen to the challenge, introducing a WiredTiger database engine that increases write speeds by 7-10x while cutting disk space by compressing data by 50%.

… So, while MongoDB certainly hasn't lost its edge, the performance argument is no longer as cut and dry as it once was.

Use Cases and Factors Affecting the Choice of Postgres or MongoDB

You might be thinking: so now what? Do I choose Postgres or MongoDB as the best JSON database?

The answer depends on what you want to achieve, and what you currently have in place. To help you make the right decision, ask yourself these 7 questions:

1. **What Application Are You Using?**
   MongoDB limits the number of database management commands you need to develop an application, which can be great for rapid prototyping, as well as on-demand queries and commands built by the application.

   That said, the application itself must insert meaningful data, and you might have to put a lot of effort into maintaining the software.

2. **How Much Structure Will You Need Later?**
   MongoDB is ideal for unstructured data, but if you plan to move to a mix of structured and unstructured data later on, or if you think ACID compliance might become important in the future, Postgres may work best.

3. **Are You Using Static JSON Data?**
   If you're using static JSON data and active data that's structured for SQL storage, Postgres is a good shout – its JSONB representation is efficient and allows for indexing. That said, you can use ODBC and BI integration to run SQL queries on MongoDB reporting, too.

4. **How Much Will You Need to Modify Your JSON Data?**
   If you want to modify your JSON data inside the data store, though, MongoDB will work better – it has tools for updating individual fields.

   To modify JSON fields in Postgres, on the other hand, you'd need to extract the whole document and then rewrite it back in when you've made your changes.

5. **Do You Need to Make Dynamic Queries?**
   MongoDB is perfect for dynamic queries of frequently written or read data. That's because it is designed to work with data of all different types that are changing all the time, without requiring complex transactions between objects. You're going to get good performance, even when running ad-hoc queries on small subsets of fields, contained in documents with loads of fields.

6. **Do You Need Automatic Sharding?**

The automatic sharding functionality of MongoDB is a good fit for IT environments that use multiple instances of standardized, commodity hardware (converged architectures).

7. **Can You Get the Right Talent?**
   The cost of plumping for either Postgres or MongoDB has a lot to do with whether you can easily find developers with the right skills to run it (as well as the availability and price of hosting platforms, etc.)!

   Postgres has been around longer and is included free of charge in many Linux operating systems, so it's well established. That's not to say you'll struggle to find MongoDB experts; it's now the fifth most popular database technology out there, after all.

   Just bear in mind what talent you have in-house, and who else you'll need to take on when making your choice.

Conclusion

I know, I know: you were hoping we'd have you a lot of time and hassle by telling you to go for one or the other, right? Trouble is, it's more complex than that, as this article has no doubt shown.

To make your decision, think really carefully about what you need out of your database system – and just as importantly, what you're likely to need in a few years' time. Not just in terms of storage, but also in terms of what you want to *do* with your data.

And yes, if you're already using either MongoDB or Postgres, changing track might feel like a massive pain in the neck, but believe us, you'll want to get this right, as soon as you can. As your data keeps growing and getting more complex, turning that ship around will only get tougher!