Should I Use MongoDB or MySQL?

Last modified: November 10, 2016

Dogs in the fight

MongoDB is one of the most popular NoSQL databases used today. Mongo's JSON like syntax makes it easy to learn and its denormalized document design make querying extremely fast. It's so fast that even major search engines like Baidu are migrating over. It puts the 'M' in MEAN stack. What's not to love?

Unfortunately for all MEAN stack enthusiasts, MySQL remains a formidable opponent. Its been around since 1995 and is still valued for addressing things like data integrity, transactions, etc. that MongoDB can't match.

Diving deeper

The decision on whether to use MongoDB or MySQL boils down to SQL vs NoSQL. While Mongo remains one of the most popular NoSQL databases, don't forget that Apache Cassandra and CouchDB are also widely used in the development community. As to whether a SQL or NoSQL is right for your app, it's important to remember a few key questions.

How organized is the data?

If the data you are working with is highly relational, it may be smart to use SQL. SQL is more normalized, meaning data is stored in tables that relate to one another. The 'join' clause makes it easy to pull shared data from related tables.

MongoDB allows for normalization techniques, however is much slower. If you reference a document from a different collection, mongoDB has to separately

retrieve another collection and then compare.

If the data structure won't be changing much and you are emphasizing data integrity, then SQL wins. SQL relies on a pre-defined schema. This allows for more consistency and control over the data that’s being stored

Conversely, if you're anticipating frequent changes to your data structure, MongoDB wins. With Mongo, you can define and store data objects on the fly. Mongo doesn't care about data types, so you can store anything from Arrays to strings within a document. This makes adding attributes on the fly a breeze. You won't have to worry about jeopardizing your whole schema when things change on the back end.

How much data will I be working with?

SQL based databases have proven efficient in working with larger data sets. However the denormalized nature of NoSQL provides inherit scalability, as document look-ups can be performed in one request. Unlike Mongo, SQL relies on multiple requests to join and retrieve related data.

What will I be doing with the data?

If you're frequently querying the database, NoSQL may be a better option (especially if your working with big data). As already mentioned, it is the denormalized, document structure of NoSQL that makes single lookups extremely fast.

Data processing is a different story. SQL transactions remain superior in providing the ability to update multiple tables at once. With transactions, if one table updates but the other fails, the whole operation is called back. With NoSQL, updating multiple documents at once is a bit more tricky.

Conclusion

While developers can argue SQL vs NoSQL indefinitely, MongoDB may be the best decision moving forward. SQL may have a more established presence in today's community, however Mongo is slowly proving more flexible and performant.

Even the aforementioned pitfalls of NoSQL are being addressed. Libraries like Mongoose are providing object modeling for Mongo, giving it a more schema like design. Documents can be referenced and populated from other collections through Mongoose. Mongoose also allows for pre save operations, making it possible to update other documents (like SQL transactions).

For such reasons, it can be argued that developers should use NoSQL solutions (such as MongoDB)moving forward.