KuppingerCole Blog

by Alexei Balaganski | February 2022

SQL vs NoSQL? Why not both?

The debate over which type of database is superior – the traditional relational (SQL) or modern document (NoSQL) ones – has continued since the NoSQL concept itself emerged around 2009. Back then, a popular opinion was that relational databases with their rigid data schemas and very limited horizontal scalability were already on the way out and alternative schema-less database architectures with their flexibility, ease of use, and seemingly limitless scalability were just the right tool for modern, highly distributed, cloud-native applications.

Alas, the enthusiasm of those early years turned up to be overly optimistic – over a decade later both SQL and NoSQL solutions are going strong without a clear winner. Personally, I have always been on the fence with regard to this argument – I firmly believe that the original dichotomy is largely artificial, caused by technology limitations that have long been resolved, and, last but not least, is completely meaningless from the business perspective.

In a sense, this reminds me of a similar situation with Big Data. Do you remember when you had to invest in a complex and mind-bogglingly expensive hardware and software platform to be able to manage a few terabytes of data? Nowadays, the only thing you need is a credit card and a public cloud account. The very notion of Big Data has almost completely disappeared already – from a modern business perspective, it's just "data".

The same considerations apply to the SQL vs NoSQL debate: as long as your corporate data is available, consistent, and reasonably protected from misuse, your applications meet certain performance criteria and can be quickly adapted to changing business requirements, and your IT costs remain within the budget, the LOB people couldn't care less about the name of your favorite database. Unfortunately, under the hood, the situation is a bit more complicated.

For people working in cybersecurity, one of the daily mantras is "Complexity is the worst enemy of security." This is true for every kind of IT infrastructure, but perhaps nowhere can making an early mistake lead to catastrophic consequences than in data(base) management. Sure, having a selection of **over a dozen specialized database engines** to choose from for your new project might sound like a great idea at first. Yet, sooner or later, you will inevitably stumble upon an unexpected limitation and face a difficult decision: should you refactor your application to support a different, more suitable

engine, or should you add another one into the mix: for example, to solve a scalability challenge or simply (ha-ha!) to be able to run business analytics on live data?

Unsurprisingly, modern database vendors think about these challenges all the time and constantly work to address them for their customers, preferably in advance. And, even less surprisingly, these developments lead to a convergence of SQL and NoSQL databases towards a universal multi-model data management platform.

Take MongoDB, the **most popular NoSQL** database platform. Over the years it has become such a de facto standard for NoSQL that other document database engines like Amazon DocumentDB and Azure Cosmos DB implement MongoDB APIs to allow existing apps and tools to connect to their services instead. Developers that rely on this compatibility layer get more flexibility in choosing a more suitable (better scalable, more integrated, or simply less expensive) alternative to MongoDB. The downside of this approach, of course, is that not all functions of the original are implemented, and the degree of compatibility can vary a lot across different projects. An even bigger challenge, however, is making NoSQL instances available to 3rd party tools that expect relational data – the most common use case being business analytics. Connecting existing BI tools to MongoDB is possible but requires additional infrastructure to set up and operate and most often does not provide access to real-time data, resulting in analysis of stale data.

Well, the more interesting is the latest news from Oracle – the database vendor known for not one but two of the most popular database engines in the world. The company just announced its own alternative for MongoDB, aptly named Oracle Database API for MongoDB. But wait, you're probably saying, isn't Oracle Database a relational database? How can it possibly emulate the capabilities of a document database like MongoDB?

Although its origins are firmly in the RDBMS camp, over the years Oracle Database has evolved into a multi-model data management platform, with native support for JSON data model available since 2014. Many customers have been using Oracle for their NoSQL projects for years, but now, with the new compatibility API, this functionality can be transparently available to any existing MongoDB-compatible application without any changes to its code. The announced API is compatible with MongoDB 4.2 and offers support for transactions and load-balanced connections. According to Oracle, the only thing developers need to change in their app is the database connection string.

But wait, you're probably asking again: how exactly is it different from other compatibility APIs mentioned above? Well, this is where the most interesting things begin… As a proper multi-model database, Oracle Autonomous Database can transparently convert document collections to relational data structures and vice versa. This means that it is possible to import an existing MongoDB database into Oracle, continue running your existing app without any modifications, **and** get all the benefits of SQL-based business analytics on live data. The opposite is possible, too: you can expose existing relational data as MongoDB collections – for example, to make analytics reports available back in the original app with no effort.

The security and compliance benefits of this approach are also worth mentioning. Since all data previously spread across disparate data sources can now reside within a single database instance, it is no longer exposed to various threats and risks during the ETL process. On the contrary, all the

enterprise-grade security features of the Oracle Autonomous Database ensure consistent protection and compliance without the need to configure or tune them manually. Oracle's MongoDB API is also backed by Oracle's native user and role management, simplifying overall access management and governance.

So, where's the catch? Well, the only one I can think of is that at the moment, the MongoDB API is only available as a part of Oracle Cloud. There are no technical limitations that would prevent it from working on-prem as well, but the company is naturally planning to use this capability to further promote its cloud services. However, considering that Oracle's solution claims to overcome several key technical limitations of both the original MongoDB Atlas service and their competitors' alternatives while remaining more economical to use, migrating your NoSQL workloads to Oracle Cloud might not be such an unthinkable idea at all… In fact, it could be just what developers tired of moving data from one database to another and dealing with complex manual processes are looking for.

The Future of Information Security – Today

KuppingerCole supports IT professionals with outstanding expertise in defining IT strategies and in relevant decision-making processes. As a leading analyst company, KuppingerCole provides first-hand vendor-neutral information. Our services allow you to feel comfortable and secure in taking decisions essential to your business.

KuppingerCole, founded in 2004, is a global Analyst Company headquartered in Europe focusing on Information Security and Identity and Access Management (IAM). KuppingerCole stands for expertise, thought leadership, outstanding practical relevance, and a vendor-neutral view on the information security market segments, covering all relevant aspects like: Identity and Access Management (IAM), Governance & Auditing Tools, Cloud and Virtualization Security, Information Protection, Mobile as well as Software Security, System and Network Security, Security Monitoring, Analytics & Reporting, Governance, and Organization & Policies.

For further information, please contact **clients@kuppingercole.com**

KuppingerCole Analysts AG
Wilhelmstraße 20-22
65185 Wiesbaden | Germany

Phone +49 (211) 23 70 77 – 0
Fax +49 (211) 23 70 77 – 11
www.kuppingercole.com