Publication date:
23 Feb 2022
Author:
Bradley Shimmin, Chief Analyst, AI & Data Analytics

A database by any other name? Oracle rolls out API compatibility for MongoDB atop its own multi-model database

Omdia view

Summary

Oracle has officially shipped an API atop Oracle Cloud Infrastructure (OCI) that is compatible with applications written for the extremely popular MongoDB NoSQL database. With this API, enterprise app developers can now seamlessly tether their MongoDB JSON code directly to Oracle's Autonomous JSON Database, which begs an important question: are distinct, purpose-built databases destined for the dustbin of history?

Too many databases

Not so long ago, in 1597, a playwright named William Shakespeare penned a few words that have followed us down through the centuries.

"What's in a name? That which we call a rose by any other name would smell just as sweet."

- Romeo and Juliet, Act 2 Scene 2

In this soliloquy by the star-crossed lover, Juliet, she hits directly at the importance of—and danger in—naming things, lamenting that if only her Romeo were known by any other name, then they wouldn't need worry about creating a bloody feud between their families. In other words, names are not the thing itself but only a pointer to that thing. Fast forward 400 some-odd years, and here in 2022 we find ourselves facing a very similar question regarding the nature of the very thing that powers every single enterprise, namely operational and analytical databases.

For the past five decades the database market has operated uniquely. Unlike markets like security firewalls, where a firewall is a firewall, the database market isn't made up of "database" vendors, but rather vendors providing one or more purpose-built databases, each tackling extremely disparate needs. For this reason, all enterprises commit time, money, and expertise to the act of maintaining many, many databases; in-memory databases for stock purchase transactions, graph databases for fraud detection, data lakes for predictive and prescriptive analytics, and document databases for client-facing applications of all shapes and sizes.

Maintaining numerous databases like these imposes a tremendous amount of technical debt by dumping data into many incompatible data silos. When Omdia asked enterprise practitioners to rate their organization's progress exploiting the value of data across the business in support of digital transformation, only 16.2% said that they had reached their goals (See IT Enterprise Insights 2022). A larger percentage (23.5%) of respondents had yet to start or were just getting underway in exploiting data across the business.

When one database is at once many databases

What's wrong with this picture? In a word, complexity. And unfortunately, moving to the cloud won't help beyond simplifying the deployment and management of a continuing menagerie of technologically inharmonious databases like PostgreSQL, MongoDB, Snowflake, Amazon Redshift, Oracle Database, IBM Db2, and Redis. But what if companies could standardize on one, single database platform itself capable of supporting or "manifesting" several disparate database types, relational, streaming, geospatial, graph, et al.?

Just a few years ago, this idea of a multi-model database that could support disparate workloads, even down to the product-specific, API-level was widely dismissed as a techno-pipedream that sounded good on paper but suffered in delivering the performance, scale, and functionality found within each unique database offering. Fast forward to today and of the 359 databases tracked and categorized according to popularity by the independent site, DB-Engines, eight of the top ten databases in the world can be considered multi-model databases. Of those, the top two databases are Oracle Database and Oracle MySQL Database. Of those two, Oracle Database supports document, graph, and spatial database models. The same goes for the next most popular database, Microsoft SQL Server.

Generally, these databases have focused on supporting the basic database structures required to support these disparate models. But increasingly, vendors such as cloud giants Oracle, Microsoft, and AWS have been seeking to take on a more targeted approach, directly supporting specific API-level functionality found within popular databases. The most apparent example of this can be found with the wildly popular NoSQL JSON document database, MongoDB, which has become the gold standard choice for developers building cloud-friendly, mobile applications. While MongoDB itself is shifting to the cloud, these vendors are doing their best to bring their own iteration of MongoDB to their cloud platforms, at least in terms of compatibility—if not functionality and performance. For example, AWS offers Amazon DocumentDB with MongoDB compatibility that directly supports MongoDB 3.6 and 4.0 APIs.

Much more than API compatibility

On its technical blog, Oracle announced Oracle Database API for MongoDB, which follows AWS' lead, enabling developers to interact with JSON document collections in Oracle Database using native MongoDB commands, compatible with MongoDB 4.2 APIs. It works by translating the MongoDB wire protocol into SQL statements. These in turn are executed by Oracle Database, which lets developers continue to use the tools, frameworks, and drivers that they're already using without modification. That means they don't have to refactor their existing code if they choose to migrate their database from MongoDB to Oracle's fully managed, cloud-native Autonomous JSON Database.

Clearly, Oracle is taking direct aim at MongoDB's own push to the cloud, offering a more mature, autonomous alternative to MongoDB Atlas. With full support for parallel analytics, indexing, and ACID transactions, along with the ability to run multiple data models, Oracle is basically providing MongoDB developers an adaptive, mature, and cloud-native database alternative that's focused on accelerating application development and productivity. For companies already deeply rooted in the Oracle Cloud and database milieu, Oracle's solution makes perfect sense and confers a number of important benefits. First, and foremost, it allows users to literally manage two disparate databases using only one provisioning, management, and security architecture. For example, users can confer the same user roles and controls across both database models, even though on their own they function very differently from one another.

Second, Oracle is in essence unifying both transactional and analytical data within one database as users can point either SQL (for analytics queries) or MongoDB API calls at the same underlying JSON data. This means that users can run full analytics queries against NoSQL data using SQL just as though it were regular relational data. Conversely, users can expose relational data alongside analytical queries as regular MongoDB collections (JSON's equivalent to a relational table).

Additionally, this unification creates a number of interesting opportunities that would otherwise remain out of reach of users working across separate relational and NoSQL databases. For example, it makes it easier for data scientists to run Oracle's in-database machine learning (ML) algorithms over JSON document data. It also enables companies to run full ACID transactions on JSON documents without worrying about imposed limits on transaction duration and data volume.

Perhaps most importantly, it allows companies to make use of the numerous cloud-native capabilities found within Oracle's cloud database service, Autonomous Database. Those capabilities include automatic and fully elastic scaling (including scale to zero and separate scalability for compute and storage), end-to-end and automated database management covering patching, upgrades, backup, recovery, as well as automated security threat detection and remediation. In addition, companies can make use of several OCI and Oracle Autonomous Database ancillary products such as Oracle Analytics Cloud, Autonomous Data Guard (for data protection and disaster recovery) and Oracle APEX service for low/no-code application development. Such synergistic capabilities, multi-model support, and fully autonomous operations are not yet the norm with purpose-built databases.But soon they will become the norm. And that is the key. Database vendors are not adding administrative complexity, nor are they removing the ability to handle disparate data types. As with the automotive industry's move toward cars capable of combining combustion and electricity, so too are database manufacturers seeking to convert traditional databases into a more modern data platform capable of handling a wide array of use cases with very little administrative overhead.

The limits and complexities of API compatibility

Certainly, there are many inherited benefits of adopting a multi-model database such as Oracle Autonomous Database to run MongoDB apps, many of which are hidden from view, such as materialized views, in-memory column storage, and other query optimization techniques. However, API compatibility is a notoriously difficult measure to pin down. A vendor cannot simply claim API compatibility and promise that customers can migrate their data and restart their existing apps on day one.

API compatibility relies on the vendor's ability to support the current API (and keep up with new versions) as well as the degree of compatibility the vendor has achieved across all available API calls. As mentioned above, Oracle is not the first to target MongoDB Atlas with cloud-native API compatibility. To that end, neither Oracle nor AWS support MongoDB's current API versions (version 5.0). Naturally, all 3rd parties will lag behind the owner of an API. However, this game of catch up presents numerous challenges for developers who are ultimately responsible for ensuring compatibility over time. Also, AWS has been working to support MongoDB within Amazon DocumentDB since 2019, and to this date has yet to provide 100% API compatibility. MongoDB itself claims that AWS can only run approximately 33% of MongoDB code without error.

Beyond API compatibility, there are also questions of functional compatibility. Quite simply, different databases often handle similar problems in very different ways. Take MongoDB's aggregation pipeline functionality, as an example. Within MongoDB, application developers looking to speed up analytical queries can use the company's Aggregation Pipeline Builder to export a query in one of many familiar app development languages like Java, Node, C#, and Python 3. These queries (aggregation pipelines) then execute on MongoDB, automatically optimizing the flow of operations spanning data query, processing, transformation, and delivery.

For a company like Oracle that has spent quite a bit of time working on optimizing SQL queries, aggregation pipelines represent a well-trodden path. As such, the company does not yet support MongoDB's vision of aggregation pipelines (doing so is on the company's roadmap, mind you). Instead, Oracle recommends that users stick with SQL, which natively supports aggregation analytics, even over JSON data that is well defined in the ISO SQL standard. Using both best practices and AI, Oracle Autonomous Database can do a tremendous amount to optimize these queries, even if it hasn't seen them before. This is crucial in the cloud, where resource inefficiencies can drive up operating costs. Increasingly, it falls to the platform vendor and database manufacturer to ensure that customers are balancing cost and performance. With Oracle Autonomous JSON this means not just auto-tuning but also auto-scaling along with exact resource requirements as opposed to wasteful fixed shapes (e.g. broad clustering T-shirt sizing).

Why invest in a multi-model database?

Should potential customers of MongoDB-compatible databases consider such challenges and inherent differences as proof that multi-model databases, while interesting, are inherently flawed or not yet ready for enterprise-class workloads? Certainly not. API compatibility issues color every facet of software development, even within the most homogeneous of companies. And while databases like MongoDB and Oracle Autonomous Database may take very different approaches to tasks like user role security and query optimization, those differences take place one level below the plane of software development—and that's where the true value of a multi-model database really shows up.

In looking at Oracle's push to run MongoDB apps on Oracle Autonomous Database, the company is targeting what is perhaps the biggest challenge faced by companies seeking to put data to work in driving business decisions. That challenge is complexity. For enterprise buyers this complexity stems from managing multiple database platforms…and the cost and exposure to risk that comes from replicating and moving data between disparate data silos. From Oracle's perspective, investing in a single, multi-model database gets rid of a significant amount of technical debt, allowing organizations to use this single codebase to more readily build, evolve, and manage database services across the widest possible set of customers. That's why not just Oracle and AWS but also Microsoft with Azure Cosmos DB are taking aim at the current darling of the app development marketplace, MongoDB, but with very different approaches—Oracle with its multi-model databases; AWS, Microsoft, and others with predominantly purpose-built databases.

In this particular instance, Oracle, Microsoft, AWS and other cloud-native hyperscalers are hoping to capitalize on the fact that MongoDB is still transitioning to the cloud with MongoDB Atlas, which to a great degree depends upon these same competitors for a place to run in the cloud. With this rollout, Oracle only supports MongoDB on its Oracle Cloud Infrastructure (OCI), which somewhat lessens the competitive nature of this announcement, at least for now, since Oracle fully intends to bring the Oracle Database API for MongoDB to its on-premises offerings such as Exadata Cloud@Customer over time. For now, this release will open up new opportunities for existing OCI customers and serve as an enticement for customers currently managing multiple databases across disparate cloud providers.

Ultimately, for Oracle, this isn't just about reaching more customers or displacing leading competitors. Rather, it's another vital proof point in support of the company's thesis that today and tomorrow's enterprise need not many databases but one, single data platform that isn't constrained by its history but able to act in accord with whatever the customer needs. Does Oracle Autonomous Database tick all of those boxes? Of course not. Do the benefits of adopting a multi-model database fully outweigh any functionality or compatibility issues? Yes and no. That of course will wholly depend on the implementation.

Oracle believes those scales will ultimately come into balance with benefits such as minimizing data replication and movement and reducing security risks in its favor. For now, what the company wants is to show how easy it can be to buy Oracle Autonomous Database as if it were more than one database, a full-fledged relational database management system (RDBMS), or a Graph database, or a JSON database (now with MongoDB compatibility). Tomorrow this "buy only the database you want" approach is likely to extend to more database workloads such as time-series and streaming analytics. Ultimately, regardless of what database customers choose to license, the company wants to provide just one, unified underlying platform. To Oracle whether that's a NoSQL or a relational database, it makes no difference. To Oracle, a rose by any name will smell just as sweet.

Appendix

Further reading

IT Enterprise Insights 2022 (October 2021)

2022 Trends to Watch: Analytics and Data Management (December 2021)

Author

Bradley Shimmin, Chief Analyst AI Platforms, Analytics, and Data Management

askananalyst@omdia.com

Citation policy

Request external citation and usage of Omdia research and data via citations@omdia.com.

Omdia consulting

We hope that this analysis will help you make informed and imaginative business decisions. If you have further requirements, Omdia's consulting team may be able to help you. For more information about Omdia's consulting capabilities, please contact us directly at consulting@omdia.com.

Copyright notice and disclaimer

CONTACT US

omdia.com

askananalyst@omdia.com