SPONSORED CONTENT | WHITE PAPER

Developer data platforms: How to develop apps in 2024 and beyond

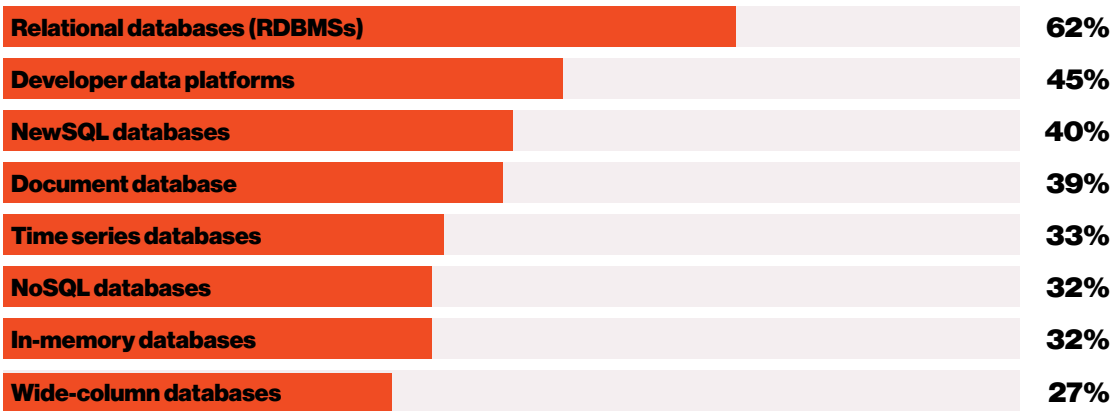
InfoWorld

SPONSORED BY MongoDB.

When developers choose a data architecture for the software they're writing, the familiar choice isn't always the easiest to work with. Yet 62% still default to relational databases (RDBMSs), according to a recent survey conducted by MongoDB and Foundry. This practice of choosing databases based on prior familiarity is antiquated and impedes efforts to continuously iterate and improve applications.

Relational databases have traditionally been used to build applications that rarely changed or evolved, but as you keep iterating and adding features, the data access patterns rarely stay the same. Relational databases are not built for frequent schema changes. Compounding the problem, relational databases use rigid structures composed of columns and rows,

What databases and platforms does your organization use in development?



Modern applications leverage multiple different data types, from geo-spatial and time-series to vector data for building AI-powered applications. Relational databases are not designed to handle multi-modal data, and bolting on niche databases increases architectural sprawl, complexity, maintenance, and speed to market. It's time to consider a different tack to shorten development time.

but modern developers use an object-oriented approach to programming.

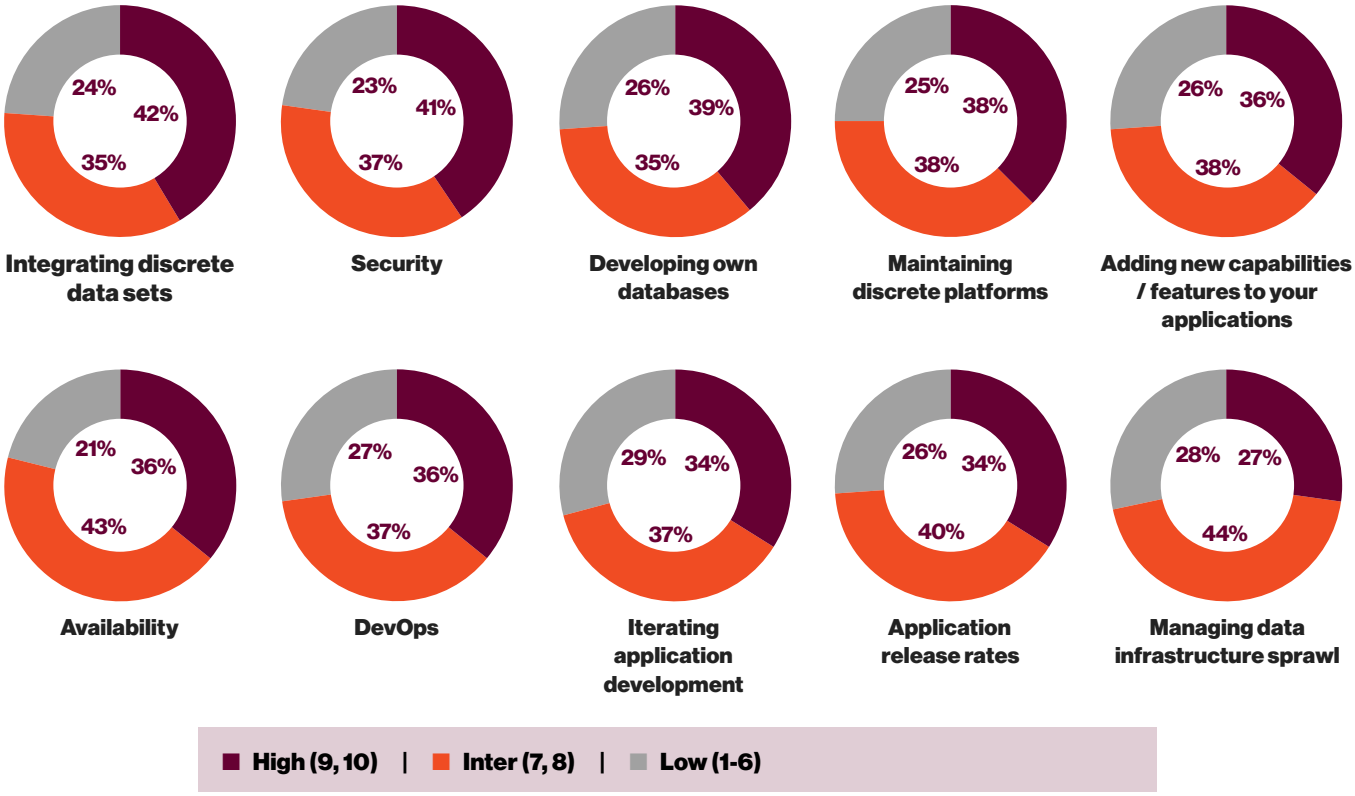
Organizations require a more flexible, familiar approach to data modeling that more accurately reflects the way developers think and code. In other words, organizations need to take another look at the flexible and familiar document database.

To paint an accurate picture of how developers are currently using databases, Foundry polled more than 100 developers across the United States. The results were clear: Many of the most common databases and platforms, including relational databases, are actually slowing software development and decreasing productivity.

The good news? Developers are beginning to understand that they need an approach for handling both structured and unstructured data, undermining the usefulness of the relational database model. This insight is made stark by the rise of generative AI (gen AI), which pulls from massive data sets of both structured and unstructured data and generates outputs predicated only on the prompt it receives in real time.

While digitalization and automation are two of the main drivers for increased speed and agility, the growth of AI-powered applications is also a major factor contributing to this trend.

How well or poorly does your organization perform on the following?

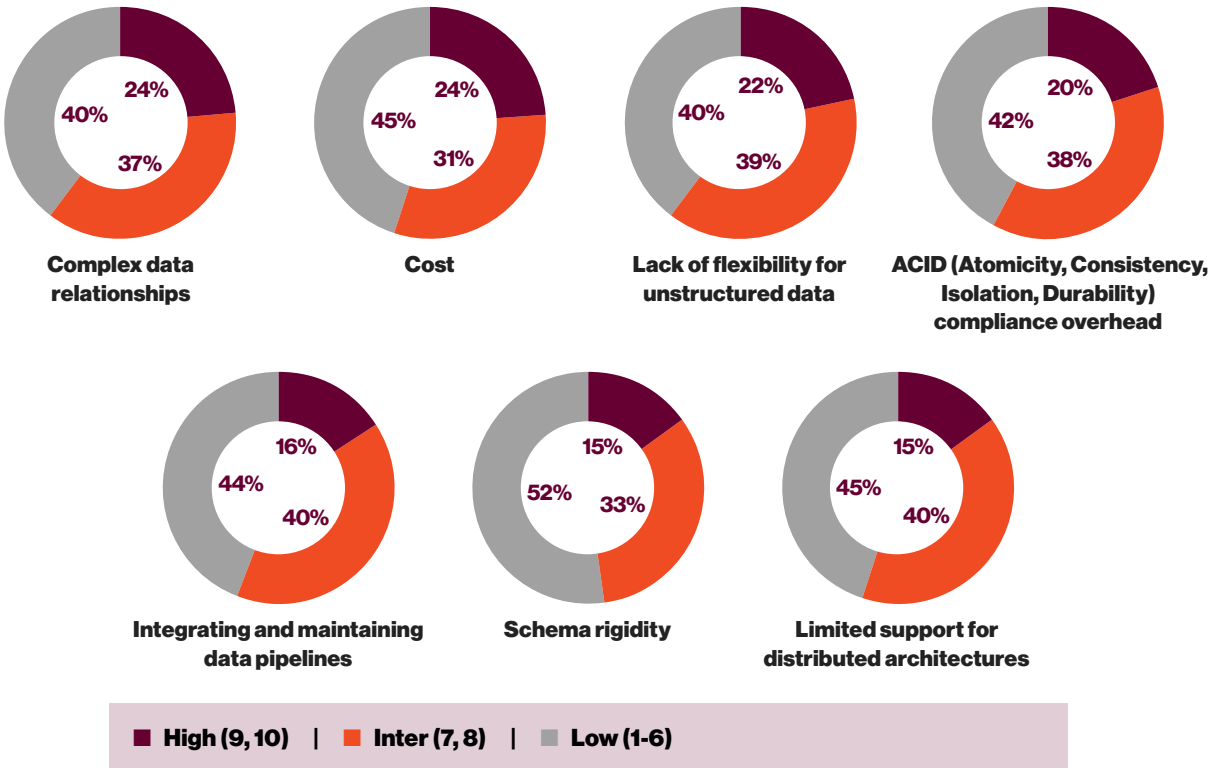


To meet this spread of demands, developer data platforms are favored by nearly half (45%) of the developers who seek greater flexibility and command in overcoming common development challenges through performance optimization, complexity management, and simplified database management.

Document databases: Powering flexibility and speed

Although relational databases tend to be the default for many developers (62% in the survey), that may be due more to habit than need. Relational databases operate on structure, consistency, and the popular SQL query language. Sometimes those are advantages for developers, but other times they are not. One big drawback is the common need for complex multiple-table joins for querying data. Most of the survey respondents (61%) find complex data relationships to be a troubling issue, more than half (55%) said cost is a problem, and 61% find the lack of flexibility in unstructured data to be a significant hindrance.

How much are developers at your organization challenged by these common limitations of traditional relational databases?



Document databases, with their dynamic and flexible schemas, are adept at managing both structured and unstructured data without the constraints of a rigid, predefined schema. This versatility makes them a go-to choice for scenarios demanding rapid and iterative development.

Developers often turn to such databases for their effectiveness in diverse areas, such as AI-driven applications; large language models (LLMs); content management systems; social networks; online marketplaces; large datasets; and agile development (DevOps). They have also become a staple for building modern applications.

Easily tailor data models to fit diverse needs

The document model excels particularly for modeling patterns for specific scenarios. This approach to data organization eschews the rigid structure of rows and columns in relational databases, opting instead for a more flexible approach within a more intuitive framework. It's particularly effective for use cases where data cannot be constrained by a fixed schema and is either hierarchical or unstructured in nature.

Applying data modeling patterns brings out the best of the document model. The extended reference pattern, for example, works effectively when your application is experiencing many repetitive JOIN operations. Identifying fields on the lookup side and bringing those frequently accessed fields into the main document improves performance. This is achieved through faster reads and a reduction in the overall number of JOINs.

Optimizing with a document database

Common limitations of relational databases — such as complex data relationships (61%), high costs (55%), inflexibility (61%), schema rigidity (48%), and limited support for distributed architectures (55%) — often make them a poor choice for developers. MongoDB and other document databases are strong alternatives that can overcome these limitations. Especially in the AI world, where developers value speed and the ability to bring together multiple data types, a data developer platform can streamline and accelerate processes. Once you understand the fundamentals of data modeling within a document database, you can optimize your model for the data

access patterns your application is most likely to use.

In summary, document databases redefine data storage and management with their flexible approach, tailored for modern applications' need for agility and speed. Coupled with the comprehensive capabilities of developer data platforms, they empower developers to innovate swiftly and effectively, making them a cornerstone of contemporary software development.

A modern alternative: Developer data platforms

In the dynamic world of large, unstructured data, document databases excel, scaling horizontally with ease. They are the engines powering rapid iterative development in domains ranging from content management and social networking to e-commerce and beyond. Their proficiency in handling large data volumes and adapting to the brisk pace of agile environments makes them indispensable in building reliable and performant applications.

Standing in contrast to the rigid nature of relational databases, document databases eliminate the complexity of multiple-table joins, offering a streamlined, developer-friendly experience. This marks a clear advantage in rapidly evolving applications.

For developers, speed is not a luxury but, rather, a necessity. The drive to accelerate application development and reduce the sprawl of data infrastructure is relentless. Developer data platforms rise to this challenge, abstracting complex data management tasks and enabling swift, efficient feature-building.

But there's more to the story than just databases. Developer data platforms extend beyond traditional database boundaries, incorporating a suite of tools and services — such as vector search, data federation, and stream processing — to facilitate comprehensive data management and analysis. These platforms are crucial for innovation, handling a range of data types — including time series, document, and graph — seamlessly integrated and optimized for any conceivable use case.

The allure of these platforms extends beyond just speed. They offer an expansive ecosystem where data stores coexist with diverse data management and analysis tools.

This synergy provides fertile ground for handling unstructured data and scaling horizontally, essential in today's data-intensive environments.

Some developers juggle a combination of different databases to optimize applications with various needs. But that strategy comes with some serious drawbacks, including increased complexity due to technology sprawl, heightened security risks, and the accumulation of technical debt. Navigating these challenges to achieve a seamless user experience requires high-level expertise in system design — or a developer data platform that can simplify this complexity.

For example, on MongoDB Atlas, a developer can easily build gen AI, time-series workloads, visualizations, event-driven capabilities, full-text search, and more on top of their data — without the need for complex integrations and with fewer lines of code.

Building new applications on developer data platforms like MongoDB is just easier; it introduces a different approach to application development. This shift emphasizes the importance of understanding

the nuances of data modeling and architecture design to accelerate the journey to a minimum viable product (MVP). To navigate this landscape effectively and explore the full potential of what you can achieve with MongoDB, we recommend leveraging the MongoDB Solutions Library. This resource is meticulously curated to provide insights and guidance on best practices, enabling you to harness MongoDB's capabilities to their fullest extent.

To successfully transition away from relational databases to a developer data platform, organizations and developers must:

- Identify the need and the goals for the transition
- Select the appropriate developer data platform and document database that align with organizational needs

- Evaluate the existing data and schema to understand the required changes
- Train the team on the new technologies and methodologies
- Migrate data and implement the developer data platform in a phased manner to ensure a smooth transition and minimal disruption

Migrate with confidence

In the evolving landscape of software development, document databases such as MongoDB represent a paradigm shift, offering a flexible, scalable, and developer-centric alternative to the traditional constraints of relational databases. Emphasizing agility and speed, these databases excel in handling unstructured data, crucial for modern applications. With the advent of developer data platforms such as MongoDB Atlas, we are at the next shift. This transition signifies a transformative approach to data management, propelling software development into a new era of efficiency and creativity.

To begin your migration journey with confidence, leverage MongoDB's bespoke relational migrator tool. This tool is designed to simplify and streamline your transition, intelligently automating major steps of the migration. Start exploring the possibilities by visiting https://www.mongodb.com/products/relational-migrator.

When you're ready to take the next step in your migration, MongoDB offers expert guidance to support you. MongoDB's specialized relational migration methodology is tailored to ensure an efficient and successful transition. Visit https://www.mongodb.com/products/consulting/relational-migration-methodology to connect with a team of seasoned experts and embark on a seamless migration to a more flexible and efficient developer data platform.

InfoWorld SPONSORED BY MongoDB®