Relational or Document:

How to Choose the Ideal Data Foundation for AI-Ready, Adaptive Systems

MongoDB.

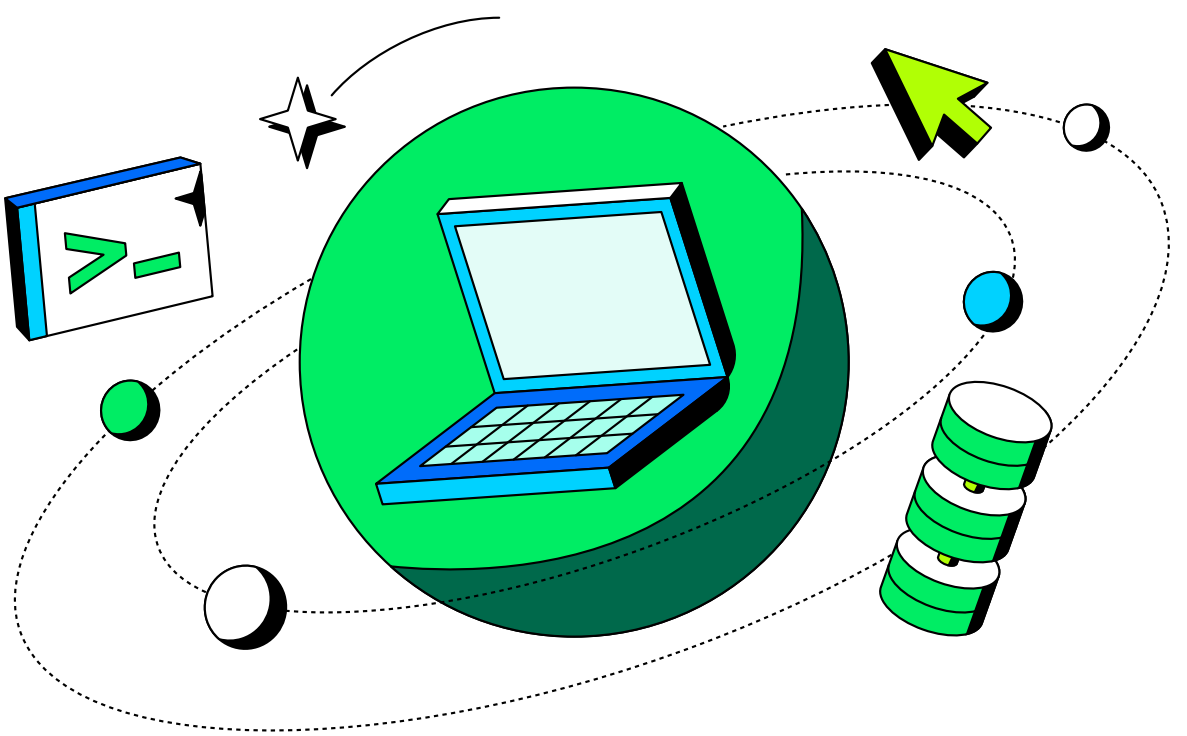
Table of Contents

Introduction: The Era of AI Acceleration

For decades, relational databases have served as the fundamental data infrastructure for organizations, supporting existing applications and benefiting from a vast ecosystem of tools and skilled professionals. Since then, the technological landscape has undergone rapid transformation. New data types have emerged, development cycles have shrunk, and the demand for always-on, globally scalable applications is greater than ever. This pace of change is set to intensify even further with the advent of artificial intelligence (AI), which is poised to revolutionize every industry, job, and task. To remain competitive, organizations must evolve at an unprecedented speed, or risk being surpassed by their rivals.

Today, organizations bring their business strategies to life through software, which powers customer interactions and product delivery. Traditional software was designed for predefined tasks and rigid routines. However, AI is expanding software's role into that of a dynamic, problem-solving partner, capable of continuous adaptation and autonomous decision-making in real-time. Software is firmly at the center of solving business problems as well as technological ones.

The ability of software to adapt is directly dependent on the speed and flexibility of its underlying data infrastructure. At the core of this foundation lies the database, responsible for storing and processing the critical data that powers all software, from managing finances to supporting intelligent agents.



The Legacy Burden: Are Relational Databases Fit for Modern Demands?

Despite their long-standing presence and historical strengths, the rigidity and inflexibility of the traditional relational model are a mismatch for today's dynamic environment. Many enterprises are hindered by the growing burden of technical debt, complexity, and inefficiency.

The Constraint of Rigid Schemas

Relational databases store data in highly structured tables with a predefined schema. This necessitates that a data model be designed and data transformed to fit this rigid structure before it can be loaded into the database. This strict adherence to schema can clash with modern code and impose complex interdependencies among engineering teams. Minor updates and schema changes become time-consuming and risky, often requiring the database to be temporarily taken offline. Anyone who has spent time working with relational models understands the dependency and approval processes involved with SQL schema changes and data migrations. Simple tasks become complex endeavors. There is also the challenge of designing relational models due to the need to split data into numerous tables and manage complex relationships, particularly when facing schema updates or combining different business entities. This rigid structure is antithetical to the adaptability required in the new progressive AI world.

Inability to Handle Diverse and Unstructured Data

A significant limitation of the relational model is the difficulty of efficiently processing unstructured or semi-structured data. This is a requirement that is increasingly prevalent in modern applications and absolutely vital for AI. For example, large language model (LLM) training demands rich document structures and potentially large binary objects that relational databases are not inherently built to handle efficiently. For today's data analytics, AI/ML, and real-time data streaming use cases, traditional relational database engines are often replaced by solutions offering flexible schemas to accommodate diverse data sources. This challenge underscores a fundamental mismatch between relational model design and the needs of modern data.

Scaling Limitations and High Costs

Relational databases are typically implemented using a "scale-up" architecture. This relies on improving performance by acquiring larger, more powerful computers with additional CPUs and memory. Such an approach becomes prohibitively expensive and eventually hits a hardware ceiling, making it difficult or even unfeasible to achieve the massive scalability required for modern, web-scale, and AI-driven applications. These systems lead to substantial technical debt, diverting engineering resources from innovation towards maintaining brittle, outmoded platforms. Engineering budgets are squandered fighting for infrastructure, refactoring applications, manually wrangling data from different systems, and stitching together point solutions.

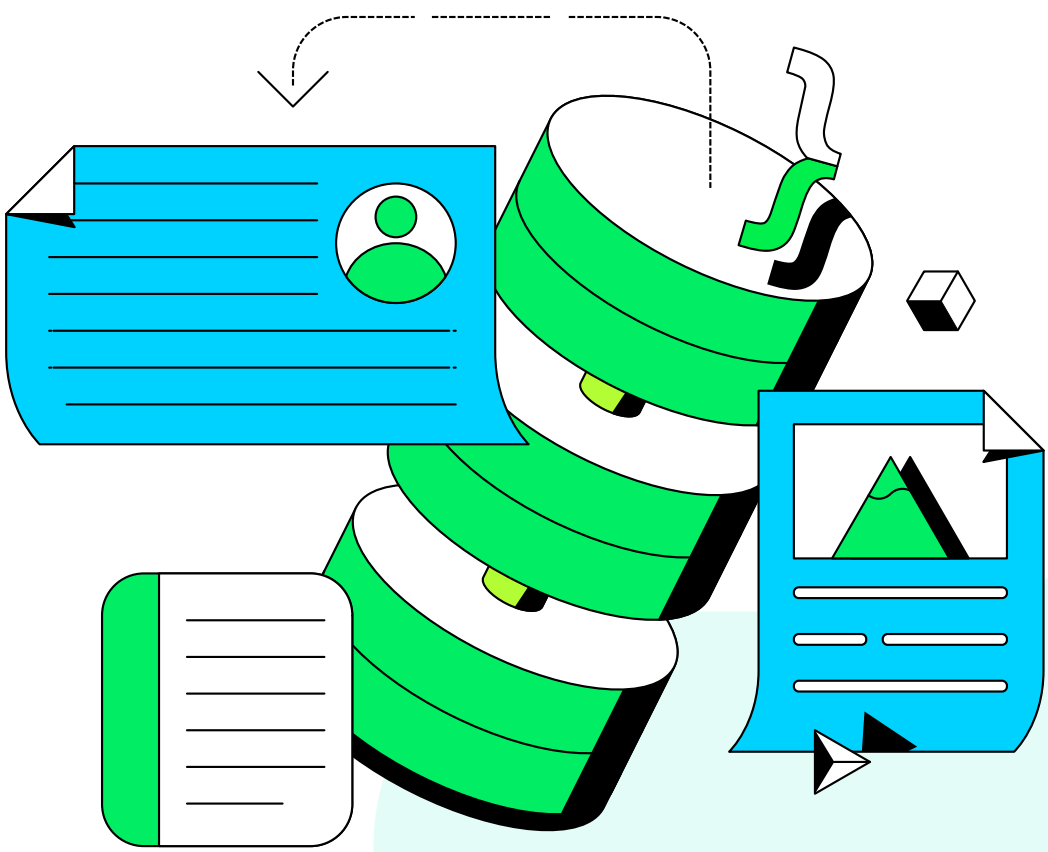
The natural solution to scaling challenges would be cluster partitioning or sharding. However, these are not native capabilities with relational databases. So, it requires developer resources and time maintaining the system rather than focusing on differentiated work.

Performance Bottlenecks and Complexity

The performance of relational databases will degrade significantly as table quantity, architectural complexity, and data volume increases. SQL databases are frequently associated with slow queries, high latencies, and timeouts, particularly when dealing with large datasets. The reliance on JOIN operations to retrieve related data from multiple tables is an acute performance impediment, creating network latency and slowing queries. Complex SQL queries involving multiple joins and aggregation functions are onerous to debug and optimize. Benchmarks have shown that JSON/JSONB attributes in relational databases introduce significant overhead for reads and writes, and off-row storage for large rows further exacerbates performance issues.

Developer Friction and Lack of AI Support

Relational databases often require copious conversions between application objects and tabular data, necessitating object-relational mappers (ORMs) as middleware layers. This rigid process slows down modern agile development methodologies. Developers spend precious time fixing SQL queries and stored procedures, detracting from innovative, differentiated development work. Furthermore, given that legacy SQL databases lack native capabilities for vector search and other AI-specific data handling, this forces a dependency on third-party extensions. Therefore, additional learning curves, and architectural and management overhead are unavoidable. This leads businesses to “bolt on” new technology to old solutions. As a result, the opportunity to adopt storage-optimized technologies built for modern workloads from the ground up slips away.



Architecting a Modern Solution: Getting the Foundations Right

NoSQL databases, and particularly document databases, were explicitly created to overcome the limitations of the previous generation of database technologies. The document approach offers flexible data models that accelerate development, especially in cloud environments, and it delivers superior performance and scalability.

Developer Friction and Lack of AI Support

The most fundamental distinction between relational and non-relational (NoSQL) systems lies with the data model. Relational approaches are bound by rigid, predefined tabular schemas. Conversely, document databases store data in documents, typically utilizing a structure similar to JSON that is highly popular among developers. The document model offers an intuitive and natural method to model data to closely align with object-oriented programming. Each document effectively acts as an object that maps to those developers work with in code.

This provides a dynamic schema where each document can contain a multitude of fields. This inherent flexibility is particularly beneficial for modeling unstructured, semi-structured, and polymorphic data. (Note: AI applications need to be able to handle diverse data forms–incuding text, video, audio, vector embeddings, and event streams–seamlessly.) Evolving an application through its lifecycle is subsequently significantly easier. For instance, new fields can be added without requiring extensive schema migrations common in relational systems. For AI models that are constantly evolving, the underlying data foundation must adapt instantly without schema overhauls or downtime. While schema design remains important for efficient querying and updating, the flexibility provided by the document model presents an advantage. This is particularly acute where application requirements are rapidly shifting and when building AI applications that learn and adapt in real-time. The document data model has broad applicability too. It can act as a superset of data models to support workloads traditionally handled by graph, key-value, time-series, and geospatial databases. This enables engineering teams to consolidate the varied data types needed by AI into a single cohesive system, eliminating fragmented silos and duplicated data stores.

Ensuring Scalability and Performance Through Scale-Out Architecture

MongoDB's design is fundamentally based on a **scale-out architecture**. This approach distributes data storage and processing work across a large cluster of commodity hardware. Capacity to expand and contract in accordance with application demand is therefore enabled due to the ability to handle massive data volumes and high traffic. This contrasts sharply with relational databases' traditional scale-up approach, which is inherently limited by the physical and cost ceilings of ever-larger machines.

This is demonstrated by companies that have benefitted from making the transition from legacy technology stacks. For example, IHS Markit, a data delivery services provider, achieved a 250x faster delivery of financial information to its customers after moving from a relational database to MongoDB.

Benchmarks[1] confirm MongoDB's (using BSON, a binary JSON format) superior efficiency for processing document data compared to PostgreSQL's JSON/JSONB attributes, due to faster binary protocols and minimal serialization overhead. MongoDB also enables horizontal scale-out without significant increases in resource consumption. This is due to its efficiency at handling massive workloads with minimal infrastructure. Features like sharding are built-in and easily configured, providing "future proofing" for data growth. This scale-out architecture also anticipates database expansion and contraction, as well as upgrades and schema changes, with **zero downtime**.

[1] Source: TheNewStack, June 2024

Maintaining an Enhanced Developer Experience and Productivity

Adoption of NoSQL databases, particularly MongoDB, has been primarily driven by developers who find it easier to create applications compared to using relational databases. MongoDB prioritizes the developer experience, providing idiomatic drivers for popular programming languages that align with how developers prefer to work, using language-specific features for efficiency.

Storing data in native formats (e.g., JSON for document databases) means developers do not have to adapt data or use complex extract, transform, and load (ETL) systems to shoe-horn semi-structured data into rigid row and column formats. This reduces transformations, extends development cycles, and means fewer applications to develop or buy to launch a new database. Developers typically find it easier to visualise documents as objects in JavaScript. This enables those teams to build quickly without initial schema concerns, defining structure later where required. MongoDB's vibrant developer community also ensures an ecosystem of additional tools and support is at hand.

Why Simplified Data Access and Aggregation is Essential

Document databases have evolved to provide rich querying capabilities often surprising to those coming from a relational heritage. MongoDB allows the querying and updating of any field in a document, supporting a comprehensive set of indexing options (including text, geospatial, compound, sparse, and unique indexes) to optimize various query patterns. This surpasses the limited querying by primary key that is offered by simpler NoSQL stores.

The document model also **reduces the need for costly and complex JOIN operations**, a significant performance impediment when deemed necessary. MongoDB's "single collection pattern" enables related documents to be stored together and retrieved in a single query, eliminating the performance penalty of cross-node JOINs. The Aggregation Framework in MongoDB simplifies complex queries by breaking them down into discrete, independently testable pipeline stages. This represents a more efficient workflow over set-based, opaque operations employed in legacy approaches. The Aggregation Framework enables real-time analytics directly against data in the database, without requiring replication to separate systems. MongoDB also offers native capabilities like graph processing, consolidating diverse application requirements into a single platform.

How Native Support for AI and Advanced Capabilities Eliminates Duplication and Unnecessary Overhead

MongoDB is an AI-ready data platform, offering native support for vector search, semantic retrieval, and real-time operational queries. Thus the need for bolted-on solutions is eliminated. MongoDB's document data model, built on a native JSON-like structure, naturally supports agentic AI. These systems often rely on standardized JSON schema protocols for output production and tool integration. This inherent compatibility streamlines data interchange and delivers a robust foundation for rapidly evolving AI workloads.

One of the world's leading healthcare companies, Novo Nordisk, uses MongoDB to support retrieval-augmented generation (RAG) with the goal of dramatically accelerating how quickly it can get new medicines approved and delivered to patients. The company's mission is to drive change to defeat serious chronic diseases. Tobias Kröpelin, NovoScribe Tech Lead and Statistical Programming Specialist at Novo Nordisk, leverages MongoDB Atlas Vector Search partly because it's LLM-agnostic. "Each foundation model has its own strengths and weaknesses, so we typically experiment with a variety of different embedding and generation models for each report we compile," Kröpelin said. "What's great about MongoDB Atlas is that we can store native vector embeddings of the report right alongside all of their associated text snippets and metadata," says Kröpelin. "This means we can run really powerful and complex queries quickly. For each vector embedding we can filter on which source document it's coming from, who wrote it, and when."

Maintaining Robustness and Consistency With a Cloud-Native Design

The relational model has accustomed developers to strongly consistent systems where writes are immediately visible. In the non-relational world, many systems have opted for eventual consistency. MongoDB however, has supported **multi-document ACID transactions** since 2018. This combines the transactional guarantees of the relational model with the flexibility and scale of non-tabular systems. Concurrently, MongoDB combines ACID transaction guarantees with **tunable consistency**. This means data can be strongly consistent by default (more familiar for developers) while allowing reads against potentially eventually consistent secondary copies when desired. Data integrity for reliable AI operations is thus assured.

While consistency is top of mind for any enterprise, resilience is just as important. MongoDB has offered built-in high-availability (HA) for over a decade and supports **zero downtime** for upgrades and schema changes, a critical feature for always-on applications. Designed for the cloud era, MongoDB Atlas offers fully managed services across major public clouds. This enables flexible deployment (on-prem, hybrid, multi-cloud) and avoids vendor lock-in. Scaling the database to expand and contract to required capacity is also handled automatically.

Conclusion: Build for Change to Drive AI-Powered Impact

In an era of accelerated change driven by AI, organizations must navigate the complexities of the digital economy and the demands of modern applications by critically evaluating whether legacy approaches remain fit for purpose. While relational databases possess historical strengths, the evolving nature of data, application requirements, and development methodologies highlight specific limitations.

Empowering businesses to innovate at the pace modern applications demand is a key tenet of MongoDB. With its highly flexible document data model, expressive query capabilities, tunable consistency, multi-document ACID transactions, and design for scale-out cloud architectures, MongoDB directly addresses the challenges posed by modern data and application development. It has continuously evolved into a fully integrated, AI-ready data platform.

Software is the engine of business strategy. An organization's software can only adapt as fast as the data infrastructure it's built upon. Legacy approaches were conceived to solve the challenges of a bygone era. The document model, as the foundation of a modern database platform, is purpose-built for continuous change. It is engineered to:

- **Handle all forms of data seamlessly:** Consolidate structured, unstructured, and semi-structured data (including text, video, audio, time series, vectors, and event streams) into a single system, eliminating fragmented silos.
- **Scale without constraints:** Adapt to unpredictable AI workloads, massive data volumes, and low-latency operations without constant re-architecting or hitting performance walls as demand spikes.
- **React instantly:** Power real-time decisions for dynamic pricing, fraud detection, and adaptive user experiences, ensuring AI systems can respond immediately to market shifts or anomalies.
- **Provide intelligent, contextual search:** Retrieve the right information at the right time, grounding AI outputs in accurate, up-to-date data for relevant product surfacing or powering intelligent agents.
- **Embed domain-specific AI:** Enhance accuracy with industry-specific AI models, leveraging vector embeddings and contextual search to reduce hallucinations and deliver more relevant, reliable, and trustworthy results.
- **Secure data at every stage:** Protect sensitive information at rest, in motion, and even in use, enabling real-time threat detection and compliance without sacrificing speed or functionality.

By reducing the friction associated with legacy models and approaches, MongoDB frees developers to focus on building applications that deliver immediate business impact. MongoDB is a strategic partner, offering both technology and deep expertise in application modernization and AI execution. MongoDB helps enterprises build and operationalize AI applications rapidly and with confidence. This ensures that organizations can design, build, and run applications that not only keep pace with but actively shape the fast-changing world.

Resources

Learn the key differences between relational and document databases in our short video, MongoDB Explained in 10 Minutes | SQL vs NoSQL. Learn their strengths and discover why developers are choosing document-based solutions.

Watch video

Start building faster with MongoDB Atlas, the fully managed cloud database loved by developers. Simplify deployment, scale effortlessly, and optimize performance. Sign up today and start exploring.

Explore MongoDB Atlas

Future-proof your applications with MongoDB's AI-powered Application Modernization Platform (AMP). Simplify migrations, harness flexible data models, and unlock innovation. Discover how to transform your apps 2x-3x faster than traditional methods.

Learn more about AMP