Use Case Guidance:

Selecting the best apps for MongoDB, and when to evaluate other options

June 2020

Table of Contents

Introduction

Data and software are today at the heart of every business, but for many organizations, realizing the full potential of the digital economy remains a significant challenge. Since the inception of MongoDB, we've believed that as companies embrace digital transformation, their developers' biggest challenges come from working with data:

- Demands for higher developer productivity and faster time to market – where release cycles are compressed to days and weeks – are being held back by rigid relational data models and traditional waterfall development practices.
- An inability to manage and extract insights from massive increases in new and rapidly changing data types – structured, semi-structured, and polymorphic data – generated by modern applications.
- Difficulty in exploiting the wholesale shift to distributed systems and cloud computing. These new platforms provide developers access to on-demand, elastically scalable compute and storage infrastructure and managed services, while meeting a whole new set of regulatory demands for data privacy.

For these reasons, non-tabular (sometimes called NoSQL or non-relational) databases have seen rapid adoption over the past decade. The problem is that these NoSQL databases do one or two things well – they might offer more flexibility in the data model, or scale-out easily. But to do this they discarded the most valuable features of relational databases, often sacrificing data quality and the ability to query and work with data in the ways needed to build rich application experiences.

MongoDB was launched in 2009 as a completely new class of general-purpose database, and quickly established itself as the most popular modern database. Architecture and development teams often get started with MongoDB in tactical projects, using the experiences they gain as a proving ground for a new class of data platform. As they start to realize the value MongoDB offers, many ask where else they can apply the technology to other projects in the business.

This white paper guides you to applicable use cases, and those workloads where you should evaluate alternative solutions.

When Should I Use MongoDB?

The MongoDB Data Platform is used across a range of transactional and analytical applications. Through its design, MongoDB provides a technology foundation that enables customers to meet the demands of modern applications.

Three core architectural principles underpin MongoDB:

1. The **document data model and MongoDB Query Language**, giving developers the fastest way to innovate in building transactional, operational and analytical applications.

2. **Multi-cloud, global database**, giving developers the freedom to run their applications anywhere with the flexibility to move across private and public clouds as requirements evolve – without having to change a single line of code
3. The **MongoDB Data Platform**, providing a unified developer experience for modern applications than span cloud to edge, in database, search, and the data lake.

With this foundation, you can address a complete range of application requirements with the MongoDB Data Platform.

MongoDB Database & Atlas: Transactional and Real-Time Analytics

The MongoDB database is general purpose, designed for transactional, operational, and and real-time analytics workloads:

- Wherever you are thinking about using a relational database, you should consider MongoDB.
- Wherever you are thinking about using a NoSQL database, you should consider MongoDB.

Whether you plan to run your applications as a serverless, cloud-native solution; in your own facilities; or a hybrid deployment model in between, the MongoDB database provides complete infrastructure agility. The best way to run MongoDB is in Atlas, our fully managed and fully automated global cloud database service available on more than 70 regions across AWS, Azure, and GCP. Alternatively you can manage MongoDB yourself on your own infrastructure with Enterprise Advanced providing a complete range of tooling and integrations to build your own private cloud.

MongoDB Atlas Data Lake: Long-Running Analytics

Beyond the MongoDB database, the MongoDB Atlas Data Lake extends the power and productivity of MongoDB to long-running analytics workloads. The Atlas Data Lake allows you to quickly and easily query data in any format on Amazon S3 (with other cloud providers coming) using the same MongoDB Query Language and tools used by the database.

With Atlas Data Lake you can realize the value of your data lake faster: you don't have to move data anywhere, you can work with complex data immediately in its native form, and with its fully-managed, serverless architecture, you control costs and remove the operational burden.

The MongoDB Data Platform

Through the MongoDB database and Atlas Data Lake, you have an integrated data platform that shares the same query language and tools, fully managed for you by the cloud-native Atlas service, enabling you to serve a broad range of transactional, operational, and analytical applications:

1. MongoDB Atlas database: designed for transactional and operational applications requiring millisecond response times, provisioned with dedicated analytics nodes for real time analytics against live, operational data (a pattern called Translytics).

2. MongoDB Atlas: Online analytics with optimized secondary indexes to support common query patterns delivering latencies in the range of subsecond to several-seconds.
3. MongoDB Atlas Data Lake: Designed for long-running analytics against large volumes of data residing in cloud storage, with latency in the range of seconds to minutes and more – dictated by how much data you are querying and how it is partitioned.

You can move data between each Atlas tier, exposing it to your operational and analytical applications as needed.

Key Strategic Initiatives Supported by MongoDB

Finding new ways to compete in the digital economy involves much more than simply inserting new technology into your application stack. Fully harnessing the opportunities presented by digital requires bringing together **people, processes,** and **platform technologies** to support your organization's strategic initiatives. Underpinned by MongoDB's data platform, we have developed a suite of solution stacks that provide:

- **Advisory consulting** to understand your strategic objectives and build a roadmap to deliver them.
- **Program governance** for delivery controls throughout a project.
- **Application lifecycle expertise** supported by design patterns and reference architectures, implementation best practices, and technical training.

From projects to modernize legacy apps, to moving to the cloud, exposing enterprise data as a service, or enabling business agility, MongoDB solutions can help you address your organization's most transformational strategic initiatives.

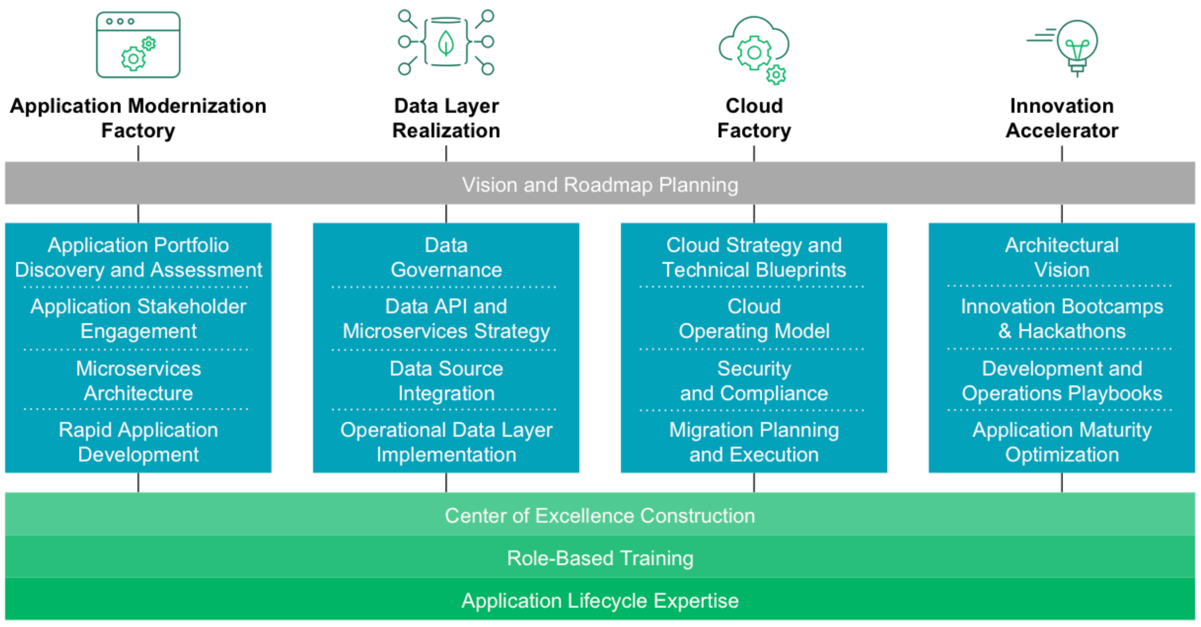
Application Modernization Factory
Data Layer Realization
Cloud Factory
Innovation Accelerator
Vision and Roadmap Planning
Application Portfolio Discovery and Assessment
Application Stakeholder Engagement
Microservices Architecture
Rapid Application Development
Data Governance
Data API and Microservices Strategy
Data Source Integration
Operational Data Layer Implementation
Cloud Strategy and Technical Blueprints
Cloud Operating Model
Security and Compliance
Migration Planning and Execution
Architectural Vision
Innovation Bootcamps & Hackathons
Development and Operations Playbooks
Application Maturity Optimization
Center of Excellence Construction
Role-Based Training
Application Lifecycle Expertise

Figure 1: *MongoDB Professional Services solutions stacks*

Legacy Modernization

Legacy Modernization enables you to apply the latest innovations in development methodologies, architectural patterns, and technologies to refresh your portfolio of legacy applications.

We work with you to build an Application Modernization Factory (AMF), cultivated from best practices developed with some of the world's largest organizations. We partner with your teams to accelerate the assessment, prioritization, and redesign of legacy apps, quantifying the economic value of change and providing a roadmap for delivery. We work with you through the modernization efforts of redevelopment, consolidation, and optimization, harnessing patterns and technologies such as agile and DevOps, microservices, cloud computing, and MongoDB best practices.

Review our Legacy Modernization overview to learn more, and download the Relational Database Migration Guide for best practices in schema design, application development, and data migration when moving from relational databases to MongoDB.

Cloud Data Strategy

Companies have long realized the agility and cost benefits of running on cloud infrastructure instead of maintaining their own data centers. However, a successful Cloud Data Strategy is much more than just using someone else's computer. To derive the agility and cost benefits that the cloud promises, you need a comprehensive approach that relies on using the right technologies and operational processes.

MongoDB is a cloud-native data platform that can meet you wherever you are on your cloud journey:

- MongoDB Atlas is our global, fully managed, on-demand cloud service for MongoDB, available on AWS, Microsoft Azure, and GCP.
- MongoDB Ops Manager provides all of the operational tooling you need to build your own private or hybrid cloud environment.
- MongoDB Realm is the serverless platform from MongoDB. It is a reliable and scalable backend that makes it simple to integrate with data and services. It streamlines development with a single syntax and data format from database to frontend app – freeing you to create better apps faster than ever.

Harnessing MongoDB's cloud services, the MongoDB Cloud Factory helps your organization take a cloud-first stance on MongoDB application development. We collaborate with your teams to develop a Cloud Operating Model, serving as a foundation for both development of new applications in the cloud and migrating existing workloads. We analyze your application portfolio to rapidly and iteratively identify applications most suitable for running in the cloud and provide technical expertise and best practices throughout the application development lifecycle.

Review our Cloud Data Strategy overview to learn more, and download the MongoDB Atlas Best Practices white paper for guidance on capacity planning, security, and performance optimization in the cloud.

Data as a Service (DaaS)

DaaS is an investment in consolidating and organizing your enterprise data in one place, then making it available to serve new and existing digital initiatives. Data as a Service becomes a system of innovation, exposing data as a cross-enterprise asset. It unlocks data from legacy systems to drive new applications, without the need to disrupt existing backends. Typical use cases include customer single view, analytics and AI, and mainframe offloading.

The path to Data as a Service is to implement an Operational Data Layer (ODL). This data layer sits in front of legacy systems, enabling you to meet challenges that the existing platforms can't handle – without a full "rip and replace" of those existing systems.

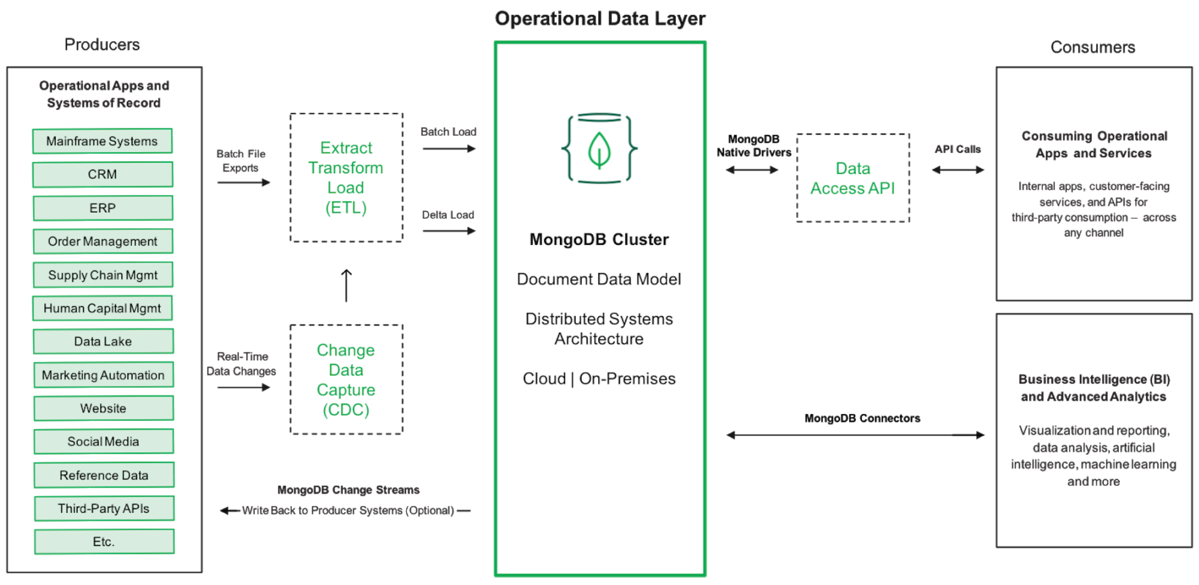
Operational Data Layer
Producers
Operational Apps and Systems of Record
Mainframe Systems
CRM
ERP
Order Management
Supply Chain Mgmt
Human Capital Mgmt
Data Lake
Marketing Automation
Website
Social Media
Reference Data
Third-Party APIs
Etc.
Batch File Exports
Extract Transform Load (ETL)
Batch Load
Delta Load
Real-Time Data Changes
Change Data Capture (CDC)
MongoDB Change Streams
Write Back to Producer Systems (Optional)
MongoDB Cluster
Document Data Model
Distributed Systems Architecture
Cloud | On-Premises
MongoDB Native Drivers
Data Access API
API Calls
MongoDB Connectors
Consumers
Consuming Operational Apps and Services
Internal apps, customer-facing services, and APIs for third-party consumption – across any channel
Business Intelligence (BI) and Advanced Analytics
Visualization and reporting, data analysis, artificial intelligence, machine learning and more

Figure 2: *Reference architecture for an Operational Data Layer*

MongoDB has developed a tried and tested approach to constructing an Operational Data Layer. The Data Layer Realization methodology helps you unlock the value of data stored in silos and legacy systems, driving rapid, iterative integration of data sources for new and consuming applications. Data Layer Realization offers the expert skills of MongoDB's consulting engineers, but also helps develop your own in-house capabilities, building deep technical expertise and best practices.

Review our Data as a Service overview to learn more, and download the ODL Reference Architecture for best practices in implementing an Operational Data Layer.

Business Agility

Organizations that are equipped to respond quickly to new opportunities, threats, and changing regulatory conditions will solidify their competitive positions and gain market share. You need to build new applications and functionality to address emerging and changing use cases better and faster than ever before.

MongoDB's Innovation Accelerator helps achieve widespread transformation of your development organization and processes to achieve faster time to market across multiple business applications and use cases. Our Professional Services team works with you to re-equip your development teams, supporting agile application development and continuous delivery of new functionality. The Innovation Accelerator:

- Establishes a roadmap for achieving business agility.
- Ensures rapid prototyping of solutions.
- Builds deep technical expertise and best practices across your development teams.
- Supports agile and iterative development, testing, and deployment.

Review our Business Agility overview to learn more.

Use Cases for MongoDB

Through the solutions discussed above, MongoDB serves a broad range of transactional and analytical applications. In each of the following use cases, we provide a definition of the application, along with required capabilities and how MongoDB can help. Review the MongoDB Architecture Guide for more detail on specific MongoDB features and capabilities applicable across all of these use cases.

Single View

A Single View, sometimes called Customer 360 or a data hub, aggregates data from multiple source systems into a central repository to create a single view of a business entity. By creating a single, real-time view, organizations enhance business visibility and enable new classes of analytics to better serve their customers and improve oversight of key resources.

While the most common use case is building a single view of your customers, the same approach can be applied to create a single view of supply chains, financial asset classes, products, and more.

Customer example: As the UK's fastest growing electronics retailer, AO.com was challenged to maintain business expansion without losing touch with its customers. It turned to MongoDB, to build a single customer view, delivering the project in just 3 months. With its single view of the customer, the business has been able to reduce call handling times by 40%, cut fraud processing from hours to seconds, and enabled its legal and marketing teams to better support new GDPR requirements.

Required Capabilities	Why MongoDB?
Data model ● Ingest data of any structure from source systems ● Dynamically adapt as a source systems schema changes	**Data model** ● Document data model to store rich, multi-structured data ● Flexible schema with no schema migrations ● MongoDB Connector for Apache Kafka to stream data changes from source systems to the single view, and to stream changes back out to consuming systems
Query model ● Multiple access patterns and query types ● Simple lookups through to sophisticated analytics for business insight and personalization	**Query model** ● Expressive query language, secondary indexing, aggregation pipeline, text search, and on-demand materialized views ● Federated queries across your Atlas database and Atlas Data Lake to access customer data wherever it is stored ● Data visualization: MongoDB Charts and BI Connector ● AI: Spark Connector, idiomatic R and Python drivers
Grow and protect ● Scale as new data sources are on-boarded ● Robust security and data sovereignty controls	**Grow and protect** ● Distributed systems architecture with native sharding for scale out ● Comprehensive access controls, encryption down to the level of individual fields, and auditing ● Global clusters to control data residence
Design and implement ● Governance processes to build and maintain the single view	**Design and implement** ● 10-Step methodology to creating a single view

Table 1: *Required capabilities for single view use cases*

Beyond technology, organizations need to implement the business processes needed to deliver and maintain a single view. Built on experiences gained over many years working with organizations of all sizes and industries, we have developed a 10-Step Single View Methodology to share best practices.

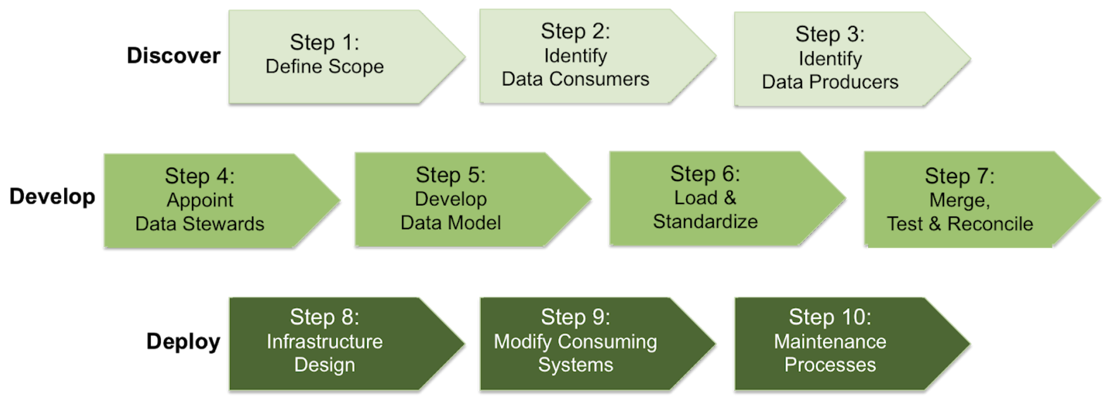
Discover
Step 1: Define Scope
Step 2: Identify Data Consumers
Step 3: Identify Data Producers
Develop
Step 4: Appoint Data Stewards
Step 5: Develop Data Model
Step 6: Load & Standardize
Step 7: Merge, Test & Reconcile
Deploy
Step 8: Infrastructure Design
Step 9: Modify Consuming Systems
Step 10: Maintenance Processes

Figure 3: *10-Step methodology to building a single view with MongoDB*

The MongoDB 10-step methodology provides a step-by-step guide to each stage of the single view project – from discovery to data loading and reconciliation, through to deployment, along with the governance and tools essential to successfully delivering a single view app with MongoDB.

Customer Data Management and Personalization

The single view use case is an example of customer data management. In this context, customer data also applies to data held on employees, citizens, patients, students, partners, through to complete corporate entities. This data is used by both backend Line of Business software such as CRM, HR, billing, and order processing, as well as front-end apps such as eCommerce user profiles and customer self-service portals.

The same set of required capabilities defined earlier for a single view applies equally to customer data management, especially in personalizing a user's experience as they interact with your apps and digital properties. To stand out from competitors, you need to offer contextual and engaging experiences that are personalized for each user – reflecting their interests and providing recommendations to them, in real time and on any device.

Customer example: Expedia is a virtual concierge – it knows if you want family-friendly holidays, business travel, or a once-in-a-lifetime break. Using MongoDB, it pushes special travel offers to users in real time by analyzing their searches and comparisons across its site.

Managing any type of personally identifiable information makes privacy a key requirement for your customer data management apps. The European Union's GDPR (General Data Protection Regulation) has ushered in a new wave of global privacy regulations that govern how organizations collect, store, process, retain, and share the personal data of citizens. You can learn more about how MongoDB can support your compliance initiatives by downloading our Global Data Privacy guide.

Internet of Things (IoT) and Time-Series Data

Today, the IoT is enabling companies to blend the physical with the digital worlds. The business value of connecting all of these "things" is realized through the creation of new revenue models, improved productivity, and the ability to generate new insights that drive operational efficiencies. The IoT already connects billions of devices worldwide, and that number is growing every day. Many market analysts predict that only by adopting IoT can organizations fully unlock the revenue opportunities promised by digital transformation.

IoT is one example of an application with time-series data at its core. Other time-series use cases include financial trading systems, clickstreams, monitoring and event logging, and asset tracking. Download our white paper to learn more about time-series schema design best practices.

Required Capabilities	Why MongoDB?
Data model ● Accommodate complex and quickly changing time-series data generated by sources such as heterogeneous sensors, connected devices, log files, and financial stock tickers	**Data model** ● Document data model to store rich, multi-structured data ● Flexible schema makes it easy to ingest new data without changes to the database
Scale, Resilience, and Data Tiering ● Ingest high volumes of sensor and event data from geographically distributed assets ● Always-on ● Cost-effectively store and access all of your time-series data, irrespective of its age	**Scale, Resilience, and Data Tiering** ● Distributed systems architecture with native sharding ● Replica sets and Global Clusters to distribute data across multiple regions ● Atlas Online Archive to automatically tier aged time-series data out of your database into your data lake for lower cost storage
Analytics ● Concurrent real-time analysis against live, operational data, without the latency of moving it to specialized storage ● Discover valuable insights with analytics and machine learning.	**Analytics** ● Workload isolation to separate data ingest from analytics processes running on the same database cluster ● Expressive query language, secondary indexing, aggregation pipeline, and on-demand materialized views ● Federated queries across your Atlas database and Atlas Data Lake to analyze time-series data wherever it stored ● Visualization: MongoDB Charts and BI Connector

	● AI: Spark Connector, idiomatic R and Python drivers
End to End Platform ● Seamlessly manage data at the edge of the network and in the core backend ● Push anomalous events to consuming systems	**End to End Platform** ● MongoDB Realm to synchronize local data at the edge with backend MongoDB Atlas database ● MongoDB Connector for Apache Kafka, change streams and Atlas triggers for reactive event driven pipelines

Table 2: *Required capabilities for IoT and time-series use cases*

IoT customer example: Bosch has built its Internet of Things suite on MongoDB, bringing the power of big data to a new range of industrial and consumer IoT applications including manufacturing, automotive, retail, energy and many others. Learn more from the case study, and review our IoT Reference Architecture.

Time-series customer example: Man AHL's Arctic application uses MongoDB to store high frequency financial services market data, handling 250M ticks per second. The hedge fund manager's quantitative researchers ("quants") use MongoDB to research, construct, and deploy new trading models in order to understand how markets behave. With MongoDB, Man AHL realized a 40x cost saving when compared to an existing proprietary database. In addition to cost savings, they were able to increase processing performance by 25x over the previous solution.

Product Catalogs and Content Management

With online and mobile sales volumes growing north of 25% year-on-year, and now accelerating in the Coronavirus pandemic, it is vital for companies to scale their eCommerce product catalogs to meet explosive demand and a tougher competitive market. At the same time, the increasing ubiquity of high speed internet connectivity and smart mobile devices is changing the content management landscape. Long gone are the days of static websites and text-based publications. They are being replaced by engaging and immersive experiences enlivened with rich media assets and user generated content, all of which have to be delivered with low latency to any device, anywhere on the planet.

While most catalog use cases are centered on ecommerce product catalogs, other examples include asset catalogs used for internal inventory management and trade catalogs used in financial services.

Content management use cases include websites, online publications, research and training materials, document management, and open data repositories. Many content management systems are custom-built, but there are also packaged platforms such as Adobe Experience Manager and Sitecore, both of which use MongoDB for data management.

Required Capabilities	Why MongoDB?
Data model ● Handle massive variability in content and catalog attributes, metadata, media assets, and UGC ● Quickly update schema as new products and content are offered	**Data model** ● Document data model to store rich, multi-structured data ● Flexible schema with no schema migrations ● GridFS to store binary assets in the database, along with simple integration to cloud object stores
User experience ● Rich queries and search as users browse the product catalog and CMS ● Never slow down under the peak loads generated by promotions, seasonal spikes, or new publications ● Never go down, always available	**User experience** ● Expressive MongoDB Query Language, Atlas Search based on Lucene, graph traversals ● Native sharding for horizontal and elastic scale in response to demand ● Global Clusters to colocate data close to users, and replication for resilience
Real time analytics ● Serve up personalized recommendations ● Monitor sales performance and content consumption in real time	**Real time analytics** ● Aggregation pipeline ● AI: Spark Connector, idiomatic R and Python drivers ● Dashboards: MongoDB Charts and BI Connector

Table 3: *Required capabilities for product catalogs and content management*

Product catalog customer example: As a top 10 global retail brand with 1.4 billion active listings across 190 markets around the world, eBay cannot afford systems downtime. This is why the company relies on MongoDB as one of its core enterprise data platform standards, powering multiple, customer-facing applications that run ebay.com. The company's product catalog is distributed and scaled on a 50-node MongoDB replica set, spread across multiple data centers.

Content management customer example: Around $4 trillion is invested globally every year in medical and scientific research. Elsevier publishes 17% of the content and discoveries generated from that research. MongoDB is at the core of the Elsevier cloud-based platform, enabling the company to apply software and analytics that turns content into actionable knowledge and new insights for its customers.

Payment Processing

With payment processing moving to web and mobile channels, organizations are seeking to modernize existing backend transactional systems to deliver the availability and scale needed to reliably serve more customers. They have to do this without giving up the strong data integrity guarantees they have come to expect from traditional relational databases.

Unlike most modern, distributed databases, MongoDB has support for multi-document transactions. Through snapshot isolation, transactions provide a consistent view of data, and enforce all-or-nothing execution to maintain data integrity. Transactions in MongoDB feel just like transactions developers are familiar with from relational databases, making them simple to add to any application that needs them. Unlike relational databases, MongoDB's transactions can operate against highly scaled-out sharded clusters.

The addition of multi-document transactions makes it easier than ever for developers to address a complete range of use-cases with MongoDB, while for many, simply knowing that they are available provides critical peace of mind. You can learn more by reading our Multi-Document ACID Transactions whitepaper.

Required Capabilities	Why MongoDB?
Data integrity ● Handle payment and order processing with data correctness guarantees	**Data integrity** ● Multi-document ACID transactions with snapshot isolation
Data model ● Support variability in payment data and financial instruments ● High precision number, temporal, and geospatial data types for lossless processing, sorting and comparisons	**Data model** ● Flexible document data model to store rich, multi-structured data ● Bank to Bank interchange standards (SWIFT, Maestro) now define JSON implementations ● Advanced BSON data types including decimal, datetime, and GeoJSON data
Analytics ● Fraud detection, risk profile and liquidity monitoring ● Monitor sales performance in real time	**Analytics** ● Aggregation pipeline with Workload Isolation, enabling translytics on the payment platform ● Federated queries across your Atlas database and Atlas Data Lake to analyze payments data wherever it stored ● AI: Spark Connector, R & Python drivers ● Dashboards: MongoDB Charts and BI Connector, with data exposed to 3rd parties via merchant portals

	● Storage tiering, discussed later in the guide to move aged data between online Atlas database and Atlas Data Lake
Resilience & Latency ● Maintain availability in the face of outages and planned maintenance	**Resilience & Latency** ● MongoDB replica sets and Global Clusters to distribute data across multiple regions
Data Integration ● Integrate eCommerce apps with payment providers ● Push payment and fraud alerts to consuming systems ● Cost-effectively store and access all of your payments data, irrespective of its age, to maintain regulatory compliance	**Data Integration** ● MongoDB Atlas triggers and functions to call 3rd party payment provides ● MongoDB Connector for Apache Kafka, change streams and Atlas triggers for event-driven pipelines ● Atlas Online Archive to automatically tier aged payments data out of your database into your data lake for lower cost storage

Table 4: *Required capabilities for payment processing*

Payment processing use cases include eCommerce backends, financial trading systems, billing engines, and mobile payment gateways.

Payment processing customer example: Cisco migrated its eCommerce platform, handling over $40bn of revenue per annum, from a legacy relational database to MongoDB. As a result, customer experience is improved by reducing latency 5x and improving availability by 100x. The migration has driven developer productivity gains, eliminating 25+ business critical backlog features. Developers can build new applications faster, while the company's eCommerce platform can tap into the business agility enabled by cloud computing.

Mobile Apps

From a "nice to have" ten years ago, mobile is now central to almost every customer engagement strategy. From smarter devices to ubiquitous sensors embedded in every-day objects, through to high performance WiFi and cellular networks, organizations embracing "mobile-first" development are opening up new opportunities.

Many organizations start their mobile journeys with mobile-optimized websites and refactored web apps such as customer portals and banking – enabling them to deliver services to customers across their preferred channels. Now new waves of mobile apps have emerged, including immersize, AI-driven lifestyle and retail experiences, mobile payments, Augmented Reality and gaming, personalized healthcare and fitness tracking, streaming services, and many more.

The MongoDB data platform underpins your mobile apps, from the device through to the backend:

- MongoDB Realm database – Radically simplifies development and improves user experience. Store data where you need it, on your iOS and Android devices to your backend in the cloud – all using a rich data model.
- MongoDB Realm and Stitch – The best way for your applications and users to access the data and services they need. Realm sits at the front end and middle layer of MongoDB's data platform, providing secure access to data hosted in MongoDB Atlas. With Realm and Stitch, you can quickly create rich, secure apps and services without app servers, web hosts, or gateways, and keep documents in sync between your mobile devices and Atlas.
- MongoDB Atlas – on-demand, elastic, and fully managed global cloud database backend, with baked-in best practices, leaving developers free to concentrate on their apps.

Required Capabilities	Why MongoDB?
Developer velocity ● Rapidly build and evolve mobile apps ● Create rich app experiences ● Feature toggles to incrementally deploy new app functionality	**Developer velocity** ● Same data model on mobile devices and backend ● Object and document data structures match objects and JSON syntax used in code ● Simple SDKs to access data stored on-device and in the backend ● Expressive query language to work on data on the device and in the backend ● Flexible data model to support multiple app feature sets
Data portability and security ● Sync data between device and backend ● Access controls to data ● Same mobile database for Android, iOS and IoT	**Data portability and security** ● Realm Sync to automatically propagate data between user devices and Atlas ● MongoDB change streams and Atlas triggers for reactive event driven pipelines ● Realm and Stitch access rules to control data visibility
Resilience and scale ● Never go down, always available ● Never slow down under the peak loads generated by new app launches	**Resilience and scale** ● Global Clusters to colocate data close to users and replication for resilience ● Realm serverless platform: auto-scale ● Native sharding for horizontal and elastic scale

Table 5: *Required capabilities for mobile apps*

Mobile customer example: 7-Eleven used MongoDB to expand customer engagement into m-commerce. Customers can browse a local store's inventory, and then click and collect product they want to buy, all via their mobile app. As well as handling the transactional needs of the app, MongoDB also layers the analytics capabilities that allow 7-Eleven to further improve customer experience.

Mainframe Offload

Despite its long predicted demise, the mainframe remains a critical IT asset in many large enterprises. But ongoing reliance on the mainframe does not come without challenges. Web, mobile, social, Artificial Intelligence, and Internet of Things applications are driving a deluge of new data. The volume, speed, and diversity of this data is overwhelming mainframe environments. Coupled with pressures to meet new regulatory demands and cut costs, CIOs are challenged in how quickly they can remake the business for digital, while trying to innovate on top of legacy technologies.

Mainframe offloading is the process of replicating commonly accessed mainframe data to an Operational Data Layer (ODL) built on MongoDB, against which operations are redirected from consuming applications. The existing mainframe is left untouched. By offloading mainframe operations to MongoDB, organizations can drive faster innovation, improved customer experience, and reduced costs.

Required Capabilities	Why MongoDB?
Data model ● Ingest system of record data from the mainframe, then enrich it with data of any structure from source systems ● Dynamically adapt as a source system's schema changes	**Data model** ● Document data model to store rich, multi-structured data ● Flexible schema with no schema migrations
Query model ● Multiple access patterns and query types over multiple channels ● Simple lookups through to sophisticated analytics for business insight	**Query model** ● Expressive query language, secondary indexing, aggregation pipeline, and on-demand materialized views ● MongoDB Realm provides simple and secure API access from clients and microservices to your data ● Data visualization: MongoDB Charts and BI Connector ● AI: Spark Connector, idiomatic R and Python drivers
Synchronize ● Ensure data is available and consistent across mainframe and operational apps	**Synchronize** ● Any data changes can be automatically propagated between the mainframe and ODL as they happen, using the MongoDB Connector for Apache Kafka

Grow and protect ● Scale as new data sources are on-boarded ● Robust security and data sovereignty controls ● Resilient, always-on	**Grow and protect** ● Distributed systems architecture with native sharding for scale out ● Comprehensive access controls, encryption down to the level of individual fields, and auditing ● Global clusters to control data residence

Table 6: *Required capabilities for mainframe offload use cases*

Customer example: Alight Solutions (formerly AON Hewitt) offloaded its Human Capital Services from the mainframe to MongoDB, improving application performance by 250x, reducing costs, and creating a platform for innovation. Review the Mainframe Offload Reference Architecture to learn more about why to offload, common approaches and patterns, enabling technologies, program execution, and governance.

Operational Analytics and AI

Every organization is striving to be data-driven. But it isn't only the data itself that is valuable – it is the insight it generates. That insight can help designers better predict new products that customers will love. It can help manufacturing companies model failures in critical components, before costly recalls. It can help financial institutions detect fraud and retailers better forecast supply and demand, while eCommerce businesses can build recommendation engines that make more informed and relevant suggestions to customers. The list goes on.

How quickly an organization can unlock and act on that insight is becoming a major source of competitive advantage. Collecting data in operational systems and then relying on nightly batch ETL (Extract Transform Load) processes to update the Enterprise Data Warehouse is no longer sufficient. Speed-to-insight is critical, and so analytics against live operational data to drive real-time action is fast becoming a necessity.

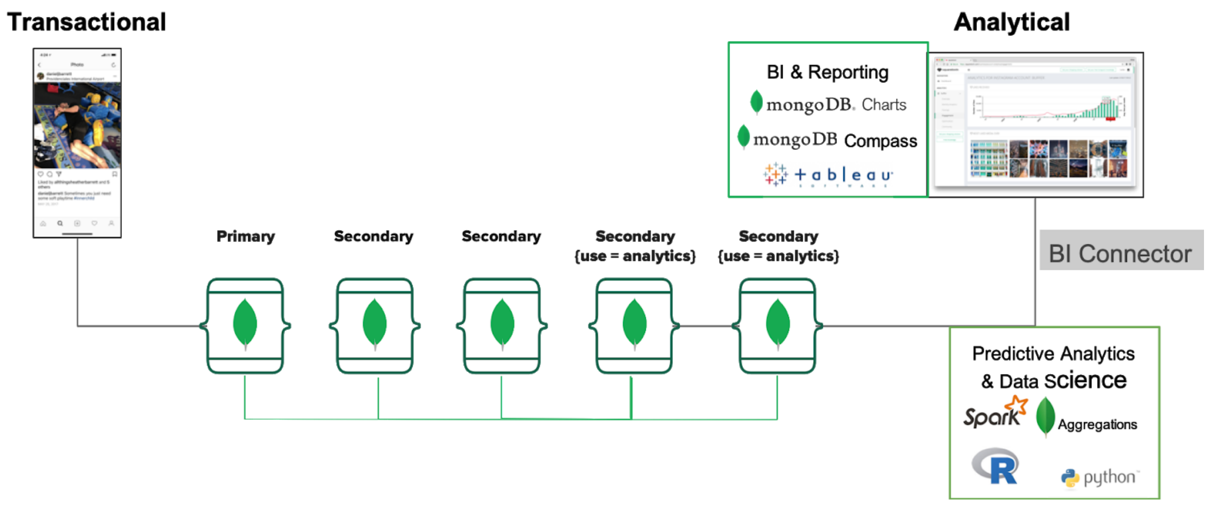

Figure 4: *MongoDB's distributed systems architecture enables the isolation of operational and real-time analytical workloads on a single cluster, providing faster and fresher insights against your data*

Required Capabilities	Why MongoDB?
Data model ● Handle data of any structure from operational systems ● Dynamically adapt the schema during machine model training	**Data model** ● Document data model to store rich, multi-structured data ● Flexible schema with no schema migrations
Online analytics ● Multiple access patterns and query types ● Simple lookups through to sophisticated data transformations and aggregations ● Eliminate fragile and lengthy ETL	**Online analytics toolkit** ● Expressive query language, secondary indexing, Lucene-based Atlas Search, aggregation pipeline, and on-demand materialized views ● Distributed architecture to enable workload isolation: separating data ingest from analytics processes running on the same database cluster
Integrated with data analytics ecosystem ● Visualize data of any structure ● Leverage investments in existing reporting and AI frameworks ● Push analytics updates to consuming systems	**Integrated with data analytics ecosystem** ● Federated queries across your Atlas database and Atlas Data Lake to analyze data and build models wherever data is stored ● Data visualization: MongoDB Charts and BI Connector ● AI: Spark Connector, idiomatic R and Python drivers ● MongoDB Atlas triggers and functions to call cloud-based machine learning frameworks ● MongoDB Connector for Apache Kafka, change streams and Atlas triggers for reactive event driven pipelines
Grow and protect ● Scale as new data sources are analyzed ● Robust security and data sovereignty controls	**Grow and protect** ● Distributed systems architecture with native sharding for scale out ● Comprehensive access controls, encryption, and auditing to protect sensitive personal data ● Global clusters to control residence of personal data

Table 7: *Required capabilities for real-time analytics and AI use cases*

Download our Deep Learning and AI Revolution Paper to learn more on why and how MongoDB is being used as the data layer to power some of the most innovative AI and analytics use-cases.

Storage Tiering for Operational Analytics

Analytical systems can grow to substantial sizes, often storing hundreds of Terabytes of data. Historically these systems were served by an Enterprise Data Warehouse where data was processed in nightly or weekly batches. However changes in customer expectations and in business requirements demand online environments that can serve subsets of the most important data in real time.

A good example for this usage pattern is the Telecom Call Data Record (CDR) store. Traditionally the CDR was used offline by internal batch business processes to generate monthly billing information and customer churn reports. Today the expectation is that customers have direct and real time access to the CDR to provide visibility into their usage patterns, calls made and data consumption, while the business wants access to the freshest data to present relevant offers, detect fraud, and analyze usage patterns.

Assuming several years of data, CDRs can quickly reach hundreds of TBs in storage size. In this scenario, tiering data can provide efficient and cost-effective data access for both the telecom provider and their customers:

- The most recent 45-days worth of CDRs represent "critical hot data", stored in memory and on SSDs to provide low-latency, real time access.
- The next three months worth of data is considered "warm" and is stored on disk. While access will not be as fast as the hot data tier, it is still delivered at an acceptable latency for user experience.
- Older data is considered cold, and can be archived to the MongoDB Data Lake running on cloud object storage where it is stored for the required seven year regulatory retention period. With MongoDB's federated query, you can access data together in the database and in the data lake with a single query, returning responses to a single endpoint.

Other examples for the tiered data pattern can be found in payment archives or payment instruction management systems which store information on the planned and executed scheduled transactions in a bank. While the last 45 days are of immediate interest to the client base, older data can be kept available in the database and in the data lake.

The key formula for system sizing in a tiered data architecture can be represented as:

Total Data Amount ** (Hot time / total time) ** ***compression ratio*** *=* ***CRITICAL DATA VOLUME*** *for hot data*

Based on customer design preference, database nodes can be created with 32GB - 64GB memory and the appropriate CPU resource. A regular 64GB node with 16 cores will hold around 5TB of critical hot data.

To illustrate system sizing, we will use an example where this approach was applied at a large European payments provider who was seeking to migrate away from their existing legacy relational data warehouse.

The customer stores 150TB of payment data for each 6-month period. Their tiering model is structured so that:

1. The most recent 30-days represents hot data that has to be queried with the lowest possible latency.
2. The next 30-days represent warm data. Latency is important, but not as critical.
3. The remaining 120-days is cold data.

Testing showed that data could be compressed by 50%, so applying the formula we documented earlier, both their hot and warm data was sized at 12.5TB each:

- *150TB * (30 / 180 days) * 50% compression = 12.5TB hot data.*

With each node handling 5TB of data, the customer was able to provision a 3-shard cluster for their hot data, and another 3-shard cluster for their warm data. Using MongoDB's Time to Live (TTL) indexes, they are able to offload all cold data into the Atlas Data Lake.

Data Lake Analytics

The public cloud is increasingly becoming the dominant ecosystem and serverless technologies are offering new alternatives to traditional data analysis and consumption. Cloud object storage, in particular AWS S3, has emerged as the new standard for durable, long-term, cost-effective data storage. At the same time, compute resources have been successfully decoupled from storage so customers can now scale each independently and on-demand.

Gaining valuable insights quickly from data stored in cloud storage, and combining it with live, operational data is a strategic goal for most organizations today but is increasingly difficult to achieve at modern scale.

With Atlas Data Lake you can quickly query and analyze data across S3 and MongoDB Atlas in its native format using the MongoDB Query Language (MQL). You can combine your live and historical data without data movement and work with complex data immediately without the need for data transformation. By leveraging a serverless, scalable query service you control costs and remove the operational burden to unlock value from your data faster.

Unlocking the Data Lake

Here is how the MongoDB Atlas Data Lake helps you get value from data in cloud storage:

- **No schema definition required.** Any richly structured data stored in JSON, BSON, CSV, TSV, Avro, and Parquet formats can be analyzed in place, with a single connection string, without incurring the complexity, cost, and time-sink of ingestion, transformation, and metadata management.

- **Work with data at any scale, anywhere.** Decoupled compute and storage allows you to seamlessly expand and contract each tier of your data lake independently. Parallelize complex queries to take advantage of existing data partitioning across multiple S3 buckets, delivering performance – even for global analytics that access data in multiple AWS regions.
- **Tier data, query it where it lives.** Tier your data across fully managed databases and cloud object storage, with the ability to query it together through a single endpoint with support for federated queries. By automatically archiving historical data, you save on transactional database storage costs without compromising on making that data queryable.
- **Use your favorite tools for accelerated productivity.** Spin up your data lake right alongside your Atlas OLTP clusters from a common UI. Then query your data using the same rich and expressive MongoDB Query Language and tools. You can query data any way you want – from simple lookups and ad-hoc queries through to sophisticated aggregation and data processing pipelines. With support for SQL you can run queries in your preferred language and connect to BI tools for data visualization.
- **On-demand and serverless**. Atlas Data Lake is completely serverless so there is no infrastructure to set up and manage, and no need to predict capacity. Pay only for the queries run and only when actively working with your data. Simply provide access to your existing S3 buckets with a few clicks from the Atlas UI, or point queries to existing archives in MongoDB Atlas, to begin exploring your data immediately.

MongoDB Atlas Data Lake Use Cases

All users across your organization can now benefit from the data in your S3 data lake and Atlas database clusters, working together to answer questions and build data-rich applications. Common use cases for MongoDB Atlas Data Lake include:

Online Archiving of data in your MongoDB Atlas Cluster: Enable sophisticated data tiering by seamlessly archiving live, operational data from MongoDB Atlas database clusters to fully-managed cloud object storage, and query it using Atlas Data Lake.

Data Lake Analytics without Data Movement or Transformation: Combine live MongoDB Atlas data together with your data on S3 without needing to move or transform it into a rigid tabular format. Eliminate the cost and complexity of data movement and quickly query all of your data as it lands, including historical and streaming event data.

Data Products & Services: Monetize internal and external data to deliver market research, data- and insights-as-a-service. Create data snapshots and time series analysis, uncover hidden trends and leverage predictive analytics to innovate faster and provide more value to customers.

You can learn more from the MongoDB Atlas Data Lake product page.

Is MongoDB Always the Right Solution?

As illustrated through this Guide, MongoDB serves many transactional and analytics use cases. But of course MongoDB will not be suitable for every workload. There may be specific requirements that are better met with alternative technologies, which we discuss below.

Common Off-the-Shelf Software Built for Relational Databases

While many organizations have successfully migrated from relational databases to MongoDB, you cannot drop-in MongoDB as a replacement for ISV packaged applications that are built around the relational data model and SQL. In these scenarios, it is better to work directly with the ISV to encourage them to offer MongoDB as the data persistence layer with their application.

To learn how MongoDB stacks up against the most popular relational databases, check out our resources pages:

- MongoDB and Oracle compared.
- MongoDB and Postgres compared.
- MongoDB and MySQL compared.

Assessing Analytics and Data Warehousing Use-Cases

It is important to understand the characteristics underpinning your analytics use-case to determine if the MongoDB database is the correct technology fit. For some, it will be, while for others more traditional data warehouses or the MongoDB Atlas Data Lake will be better suited.

First we will cover the workload characteristics that are well aligned to MongoDB, and then discuss those that are better served by other technologies.

When is the MongoDB database suitable for your analytics environment?

Much of the data that organizations need to analyze is complex and richly structured – requiring many diverse attributes to model the business domain. As the digital economy continues to expand, growing volumes of rich, multi-structured data generated by modern applications are pushing traditional analytical environments beyond their design limits, forcing users to seek alternatives. MongoDB has proven to be that alternative in many projects across a variety of industries – from banking and retail, to manufacturing, utilities, and more.

The **first important criterion** to evaluate for MongoDB suitability is the data model. Consider financial instruments used in banking systems. Each instrument, whether it's a bond, loan, cash, derivative, equity swap, etc, can require hundreds of different attributes to describe and track the asset class. Alternatively product catalogs in retail systems must manage massive diversity in product descriptions, metadata, user-generated reviews, local currencies, and so on.

Traditionally these attributes have been normalized and shred across tens to hundreds of different tables in a relational database. These tables then undergo extensive transformation as they are extracted from transactional systems and loaded into a separate analytics platform to serve long-running and complex queries. To support multiple analytics query patterns, the data is often copied into multiple table structures, each designed to service a distinct reporting requirement.

MongoDB's flexible document data model allows users to take **a different approach**. With subdocuments and arrays, users can create domain-driven data structures allowing the complete business entity to be modeled and enriched in a single document, rather than shredding it across separate records and tables. Doing this minimizes the need to JOIN multiple records together when the query is run, or to create multiple copies of the same data – each in its own distinct structure to service specific queries.

Coupling domain-driven data modeling with standard features of the MongoDB database enables you to power demanding analytics workloads at scale. Specific MongoDB features include:

- Powerful secondary indexes to optimize access patterns to the data.
- The aggregation pipeline to enrich, reshape, and filter data as needed for different query types.
- On-demand materialized views to pre-compute and cache query results.

Users can easily visualize results of analytics queries with MongoDB Charts or with the BI Connector supporting all of the leading SQL-based BI tools. One retailer was able to reduce the time taken to refresh BI views of their data from six minutes in their existing analytics environment to less than two seconds using the MongoDB features described above. The aggregation pipeline was used to filter out data that wasn't needed for the query results, and materialized views were used to pre-compute and store the result set for low latency access by multiple data consumers.

In addition to visualizations and reporting, you can also feed your MongoDB data into AI frameworks with the MongoDB Spark Connector and R or Python drivers.

Experience with existing clients across multiple industries proves that MongoDB is most suitable for complex analytics use-cases when:

1. Users can take advantage of document structures for domain-driven data modeling.
2. Query access patterns can be optimized with secondary indexes, and further accelerated with materialized views.
3. The ratio of hot/warm/cold data is split roughly as 20%/20%/60% of total data size.
4. The hot (i.e., most frequently accessed) data is 50TB or less.

MongoDB is proven on analytics use-cases matching these criteria with data sizes reaching hundreds of TBs.

When data sizes exceed the guidelines above, you can couple the MongoDB database with the MongoDB Atlas Data Lake to power analytics use-cases over larger data sets. Aged data can be automatically archived out of Atlas databases to the Atlas Data Lake. This offers the benefit of lowering storage costs for the aged data, which in turn needs to be balanced against higher latency when analytics queries operate against cloud object storage,

With Federated Query, you can submit a single query to your Atlas cluster and Atlas Data Lake S3 store, returning a single query result that blends data from both the database and the data lake.

When is it better to use a traditional data warehouse?

If your use-case prevents the use of domain-driven data modeling and queries are mainly ad-hoc, one-off analytics questions that need to scan all of your data set, then a traditional data warehouse can be a more optimal solution. In this scenario, MongoDB Atlas can serve query results from the data warehouse at low latency and high concurrency to data consumers, with Atlas Data Lake handling data structures that can't be easily or efficiently mapped into the tabular data model of the data warehouse.

Data Warehouse Replacements with Atlas Data Lake

The MongoDB Atlas Data Lake is **not a direct replacement of a data warehouse**. Rather, it is **a powerful complement** to that technology. Data warehouses offer comprehensive controls for ETL, data lineage, metadata management and cataloging, governance, and workload management, backed by an extensive ecosystem of skills and tools.

The following table identifies workload characteristics to demonstrate the different properties of traditional data warehouses and the Atlas Data Lake.

	Data Warehouse	Atlas Data Lake
Data sets	Highly curated, cleansed, filtered, aggregated data, ingested via ETL processes from operational databases and applications on-premise and in the cloud	Vast pools of raw data, stored in its native form in the cloud, ingested from logs, sensors, devices, streams, APIs, and operational databases
Data structures	Pre-defined and fixed, tabular schema, with well defined constraints and relationships	Dynamic and flexible schema, rich data structures of any shape

Query patterns	Highly optimized for specific reporting and BI purposes including dashboards, statistics and predictive modeling, regressions, and decision trees	Ad-hoc, data exploration and discovery, machine learning workloads, dashboards
Consumers	Business analysts and data scientists backed by a broad ecosystem of SQL-based tools and skills	Developers, data scientists, data engineers

The Atlas Data Lake is an ideal option for those organizations exploring modern, cloud-native approaches to harnessing value from new sources of data flowing into their business.

Conclusion

MongoDB is the most popular and widely used modern data platform in the market. Developers working with data touch more than the database, and need the best way to work with data across all their systems. Through its natural and flexible data model, expressive query language, distributed design, and platform agnostic architecture, MongoDB is able to serve the most demanding operational, real-time analytics and data lake use cases your teams are building today.

MongoDB supports its platform-based approach through an array of enabling technologies, training, and professional services. Get in touch to learn more about MongoDB for modern apps, review use cases, and more

We Can Help

We are the company that builds and runs MongoDB. Over 14,200 organizations rely on our commercial products. We offer software and services to make your life easier:

- MongoDB Atlas is the database as a service for MongoDB, available on AWS, Azure, and GCP. It lets you focus on apps instead of ops. With MongoDB Atlas, you only pay for what you use with a convenient hourly billing model. Atlas auto-scales in response to application demand with no downtime, offering full security, resilience, and high performance.
- MongoDB Enterprise Advanced is the best way to run MongoDB on your own infrastructure. It's a finely-tuned package of advanced software, support, certifications, and other services designed for the way you do business.
- MongoDB Atlas Data Lake allows you to quickly and easily query data in any format on Amazon S3 using the MongoDB Query Language and tools. You don't have to move data anywhere, you can work with complex data immediately in its native form, and with its fully-managed, serverless architecture, you control costs and remove the operational burden.

- MongoDB Charts is the best way to create visualizations of MongoDB data anywhere. Build visualizations quickly and easily to analyze complex, nested data. Embed individual charts into any web application or assemble them into live dashboards for sharing.
- The MongoDB Realm and Stitch Platform helps you build better fullstack apps faster. It offers easily configurable rules for accessing data and services directly from your application frontend, along with serverless functions to execute application logic. You can automatically sync data between the client and backend data layer. Through integrations with your code repositories you can develop locally and then seamlessly deploy version-controlled application updates to test and production.
- MongoDB Realm Database has been installed over 2 billion times, offering a fast, easy-to-use, alternative to SQLite and Core Data. With support for complex queries, safe threading, a reactive architecture to create responsive and fluent UIs, encryption, and cross-platform support, developers can simplify their code and build powerful and engaging experiences on more devices.
- MongoDB Cloud Manager is a cloud-based tool that helps you manage MongoDB on your own infrastructure. With automated provisioning, fine-grained monitoring, and continuous backups, you get a full management suite that reduces operational overhead, while maintaining full control over your databases.
- MongoDB Consulting packages get you to production faster, help you tune performance in production, help you scale, and free you up to focus on your next release.
- MongoDB Training helps you become a MongoDB expert, from design to operating mission-critical systems at scale. Whether you're a developer, DBA, or architect, we can make you better at MongoDB.

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

- Case Studies
- Presentations
- White Papers
- Free Online Training
- Webinars and Events
- Documentation

Safe Harbor
The development, release, and timing of any features or functionality described for our products remains at our sole discretion. This information is merely intended to outline our general product direction and it should not be relied on in making a purchasing decision nor is this a commitment, promise or legal obligation to deliver any material, code, or functionality.

mongoDB®