The 5 Phases of Banking Modernization

Accelerate your digital transformation while minimizing risk

Nov 2024

MongoDB.

Table of Contents

Modernizing Legacy Systems

It’s clear to banks and their customers that modern banking experiences need to deliver greater convenience in real time without compromising security.

Banking consumers and financial services companies have come to expect popular features like mobile deposits, multi-factor authentication, AI-enhanced services, chatbots, speedy dispute resolution, instant money transfers, and a range of other analytical and business insights. For banks, these services are a way to streamline operations, automate services, and differentiate themselves from competitors in a market where costs are high and margins are slim.

Although both parties are clear on what they want out of modern banking, successfully implementing these services with existing technical infrastructure is difficult and slow.

To innovate and give customers the modern banking experiences they expect, banks must first free themselves from the rigid data architectures associated with legacy hardware and monolithic enterprise banking applications.

Legacy systems like relational database management systems (RDBMS) make change harder than it needs to be, stalling innovation and entrenching a fear of failure. They also complicate business requirements that didn’t exist when RDBMS were invented, such as regulatory compliance.

Because of the inflexible formats required by relational schemas, updating applications and adding features becomes a chore. The inflexible nature of relational architectures also leads to silos where data becomes duplicated across different systems and difficult to analyze.

Despite the limitations of the relational model, organizations are hesitant to embark on a solution. Legacy modernization is frequently perceived to be time-consuming, complex, and error-prone.

The reality is that legacy modernization can be straightforward, predictable, and successful, allowing banks to accelerate their digital transformation, deliver truly modern banking experiences, and support compliance with increasingly restrictive data privacy regulations, all while minimizing risk.

The incentives for modernizing are compelling. Banks and other financial institutions that have successfully modernized have seen cost reductions, faster performance, simpler compliance practices, and rapid development cycles. New, flexible architectures have accelerated the creation of value-added services for consumers and corporate clients.

MongoDB has helped thousands of organizations refactor their legacy, monolithic systems through a straightforward and strategic plan. MongoDB's approach to modernization enables banks to modernize iteratively while balancing performance and risk through five phases.

The 5 Phases of Banking Modernization

Banking systems face a key challenge: protecting existing assets and operations while modernizing.

The operational data layer

Our approach to modernization begins with a key component called the operational data layer (ODL). The ODL acts as a bridge between a bank's existing systems and the new, refactored environment. It is typically deployed in front of legacy systems to enable new business initiatives and meet fresh requirements that the existing architecture can't handle—without the difficulty, disruption, and risk of a full rip and replace of legacy systems. It can reduce the workload and cost of source systems, improve availability, reduce end-user response times, combine data from multiple systems into a single repository, serve as a foundation for re-architecting a monolithic application into a suite of microservices, and more. The operational data layer becomes a system of innovation, allowing the business to take an iterative and progressive approach to digital transformation.

The ODL is the on-ramp for data that's being routed to the new environment. It performs the following functions:

- Centrally integrates and organizes siloed enterprise data
- Makes data available to consuming applications
- Enables legacy modernization and data-as-a-service
- Creates a single source of truth
- Enables real-time analytics and mainframe offload
- Allows for gradual refactoring (vs. rip and replace)
- Minimizes disruption when deploying to the cloud

- Serves legacy data to new applications without straining the legacy system
- Makes data immediately available for analysis and business intelligence

Gradually, more reads and writes are routed to the new environment as the legacy RDBMS is retired one step at a time. By the final phase, all applications will live in the new environment.

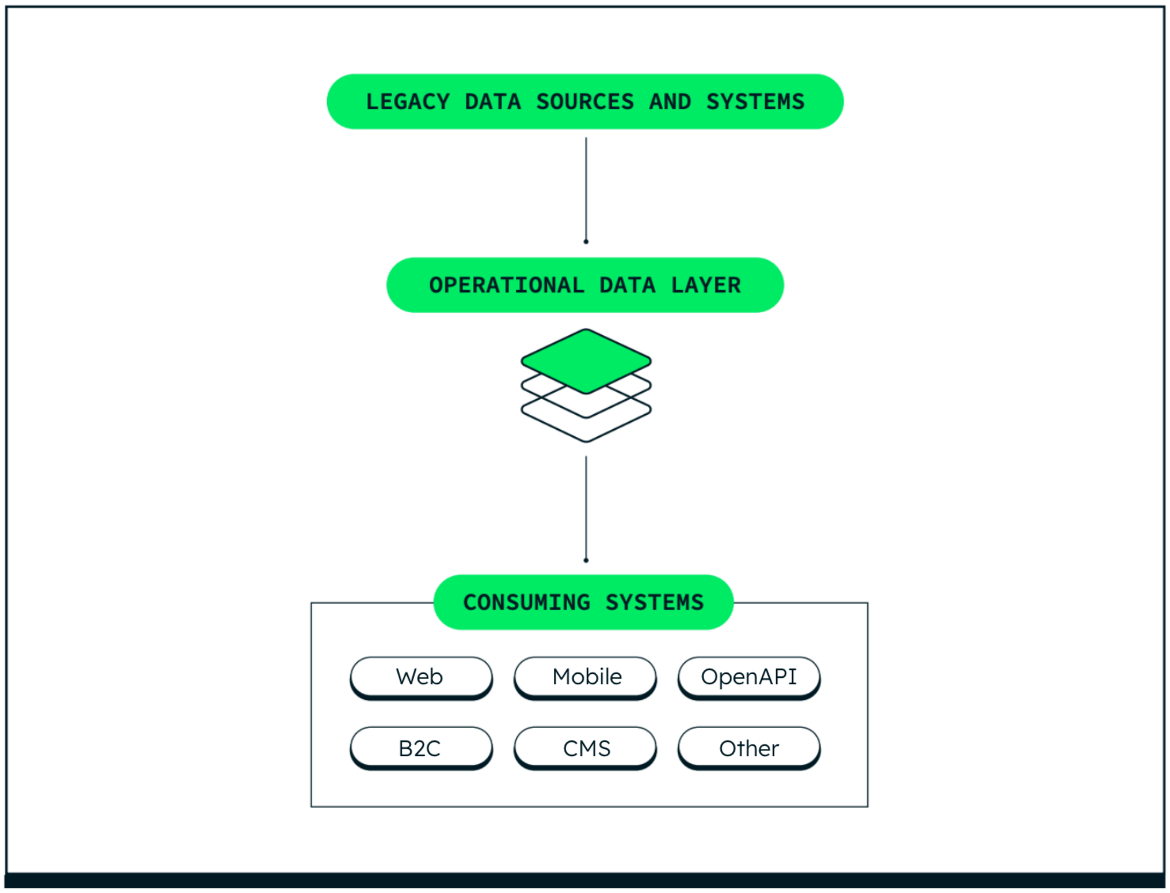


Figure 1: The Operational Data Layer (ODL)

Modern applications need operational data layers that can handle diverse workloads, scale efficiently, and enable rapid development while ensuring data security and privacy at the same time. MongoDB, with its flexible schema and robust platform capabilities, is uniquely suited for this role. Here are a few key reasons:

Flexible data model for complex applications: MongoDB's document model accommodates complex and evolving data structures. Unlike traditional relational databases, which require predefined schemas, MongoDB allows developers to model data naturally, aligning closely with the data access patterns of the application. This flexibility speeds up development and reduces overhead when changes occur, a critical factor in dynamic industries like financial services.

Real-time performance at scale: Operational workloads often require low-latency, high-throughput systems to handle transactional data. Ad hoc queries, indexing, and real-time aggregation of MongoDB provide powerful ways to access data. MongoDB's distributed architecture ensures high availability and horizontal scalability, enabling organizations to scale out easily as data grows. With features like sharding and auto-scaling, MongoDB provides the responsiveness required for operational systems, even during peak loads.

Seamless integration with modern architectures: MongoDB integrates natively with microservices and event-driven architectures, making it a natural fit for organizations embracing cloud-native and artificial intelligence (AI) applications. Its ability to act as a real-time data source for modern APIs enables seamless integration with internal applications, especially open finance ecosystems, which often prefer using OpenAI and JSON format for standardized data exchange. MongoDB's multimodal data also supports vector data management and search capabilities, enabling improved hyper-personalized user experiences, accelerating generative AI application development, and future-proofing customers' modern operational workloads.

Multi-cloud and developer-friendly: MongoDB enables customers to deploy anywhere—on-premises, in a self-managed private cloud, or as a managed service on the public cloud. MongoDB Atlas, the fully managed cloud database service, simplifies operational management, freeing teams from infrastructure concerns. With its multi-cloud offering and ability to create clusters across multiple leading hyperscalers, customers can overcome single cloud outages, achieve higher availability, and reduce cloud concentration risks. MongoDB's rich development ecosystem, including SDKs, connectors, and integration with CI/CD pipelines accelerates application development and operations, making it a developer-friendly choice.

Enterprise-grade security and compliance: MongoDB comes with a robust set of security features, including encryption at rest and in transit, fine-grained access controls, and auditing. For industries with stringent regulatory requirements, MongoDB Atlas supports compliance with multiple standards such as ISO 27001, PCI-DSS, SOC, and others, ensuring sensitive data is protected. MongoDB's groundbreaking and industry-first Queryable Encryption allows customers to encrypt sensitive application data, store it securely in an encrypted state in MongoDB, and perform equality and range queries directly on the encrypted data—with no cryptography expertise required.

The 5 phases

MongoDB's iterative approach to modernization can be broken into five phases, allowing banks to see progress toward modernization at each step along the way while still protecting existing assets and business-critical operations.

1. Simple ODL

In the first phase of legacy modernization, reads from the legacy mainframe are offloaded to the ODL. This reduces read traffic to the mainframe. The ODL provides high availability, improves performance, and handles long-running analytics queries. The ODL is interpreted directly by the application. It has a modern interface, and you can start building modern applications on the ODL.

2. Enriched ODL

In the second phase, the ODL acts as an integration layer enriched with multiple data sources and metadata. At this stage, you can begin building microservices on top of your data. The ODL also serves as an operational intelligence platform for insights and analysis. The ODL offloads more reads from the source systems and enables more use cases than were previously possible, including a single customer view.

3. Parallel write

In the third phase, reads and writes are performed concurrently on the source system and the ODL, either directly from application logic or through a messaging system, API layer, or other intermediary. This is also known as Y-loading or Y-storing. This phase lays the foundation for a more transformational shift of the ODL's role in the system architecture. In this phase, you can test the ODL to ensure functionality before using it as the primary system for writes.

4. System of transaction

In the fourth phase, transactions are written first to the ODL and then passed on to the legacy system if necessary. At this point, the ODL is the single source of truth. The secondary write to the legacy source can be accomplished with a change capture system listening to the ODL or a similar system, such as MongoDB Change Streams and MongoDB Atlas Triggers.

5. System of record

In the fifth phase, the ODL becomes the system of record for all consuming applications. The source system can be decommissioned for cost savings and architectural simplicity.

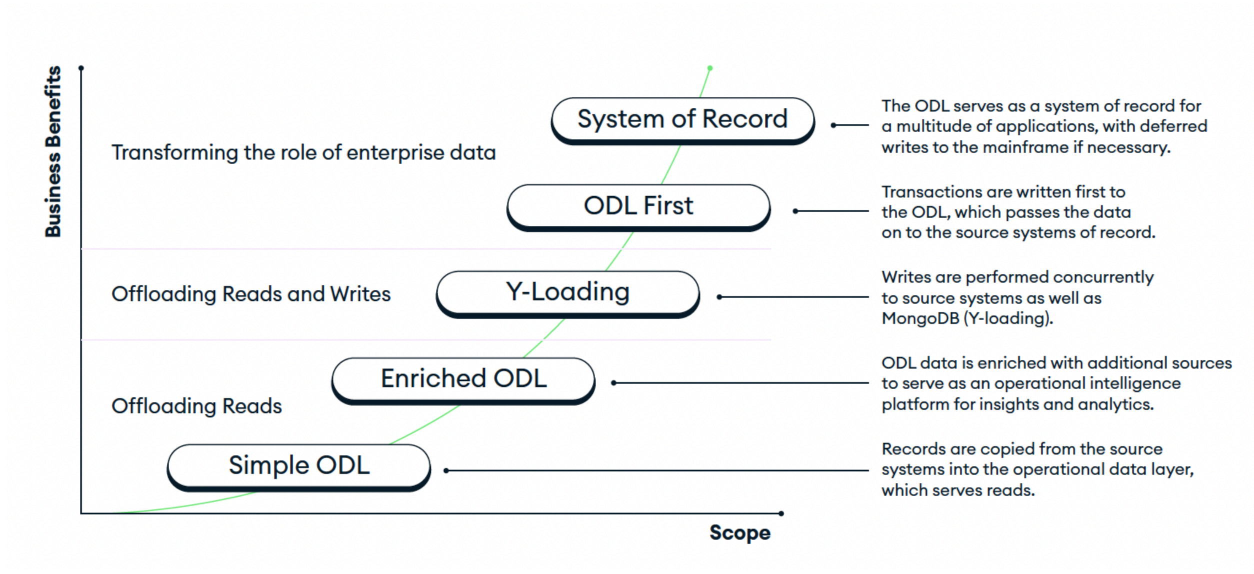

Business Benefits
Transforming the role of enterprise data
System of Record
The ODL serves as a system of record for a multitude of applications, with deferred writes to the mainframe if necessary.
ODL First
Transactions are written first to the ODL, which passes the data on to the source systems of record.
Offloading Reads and Writes
Y-Loading
Writes are performed concurrently to source systems as well as MongoDB (Y-loading).
Enriched ODL
ODL data is enriched with additional sources to serve as an operational intelligence platform for insights and analytics.
Offloading Reads
Simple ODL
Records are copied from the source systems into the operational data layer, which serves reads.
Scope

Figure 2: The five phases of banking modernization

Approaches to Modernization Planning

Before you begin migrating any data, the first step is planning your modernization. This is where you determine whether to address the data architecture first, applications first, or follow a blended approach. Each approach to legacy modernization carries its own advantages and complexities.

1. Data-driven modernization

This approach begins by moving data from the legacy system to the new environment before any applications or microservices are provisioned (Figure 3). Even in its earliest phases, data-driven modernization is a big step forward over legacy systems because once you've moved your first data source into the new environment, you can leverage it immediately and start building modern applications on top of it. Applications can write directly to the new environment without affecting the existing one. Once more writes are executed in the new environment than the old one, you can begin to dramatically reduce the footprint of the legacy system. By the time the last phases of data-driven modernization are implemented—when the new environment takes over the majority of the work and becomes the system of record—you can begin to retire legacy applications entirely.

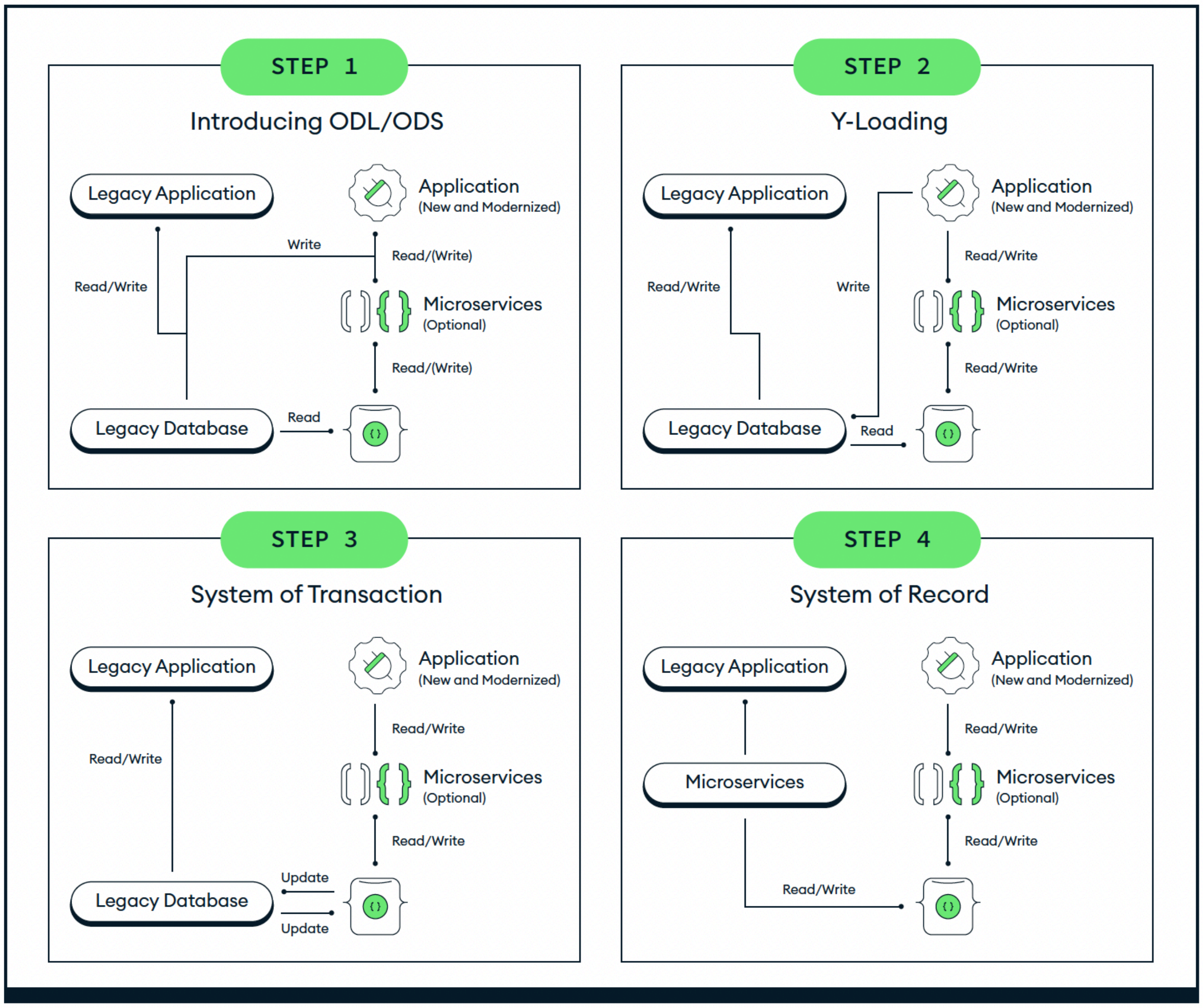


Figure 3: Data-driven modernization

2. Application-driven modernization

With application-driven modernization (Figure 4), all reads and writes from new applications and microservices are executed in the new data environment from the start. Existing traffic continues to route to the existing data store. The legacy system continues to operate unchanged. This enables new functionality to be introduced immediately, but it also introduces more complexity. Because application-driven modernization is an all-or-nothing approach, banks must have a clear strategy for retiring the legacy applications in due course.

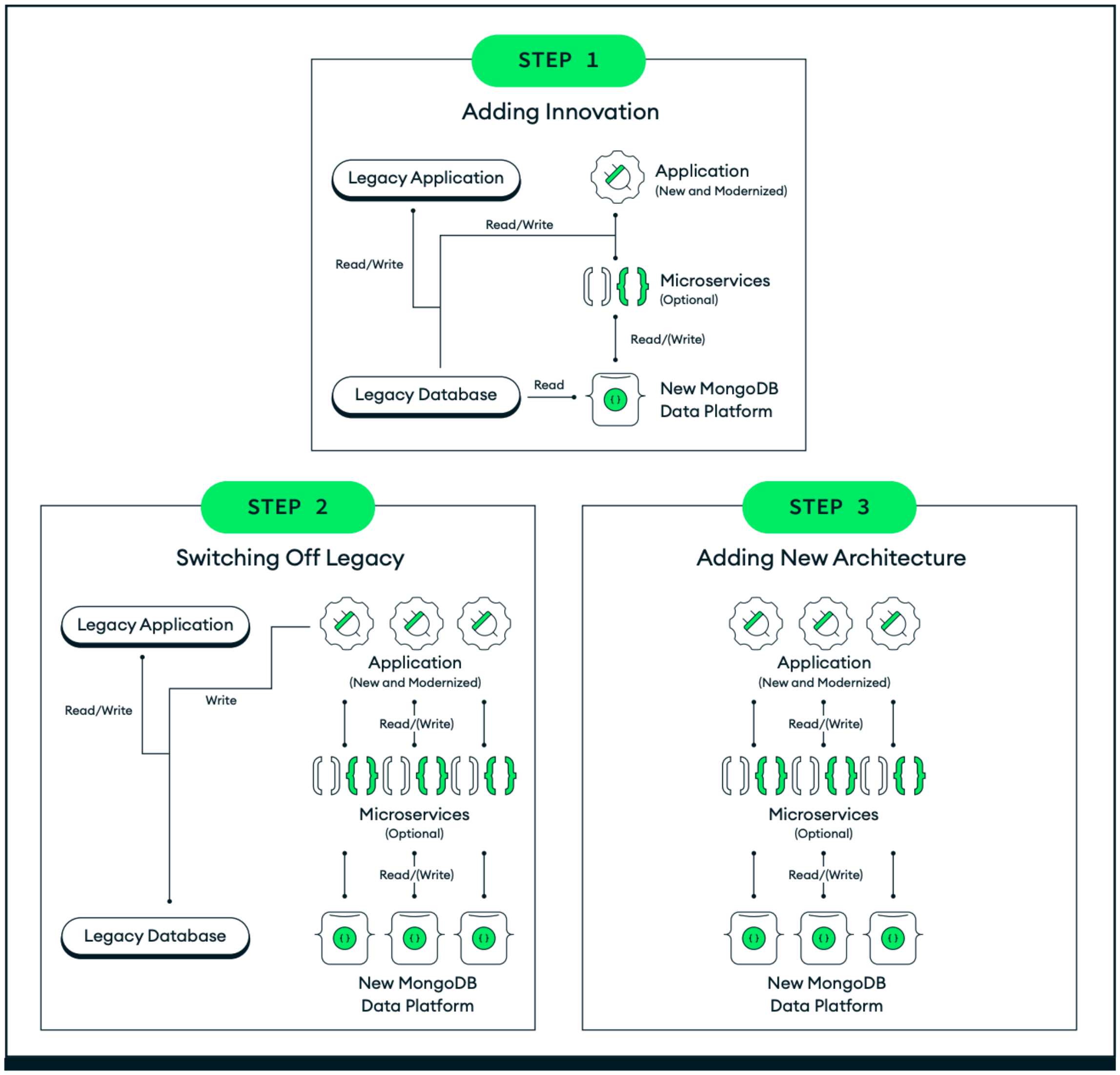


Figure 4: Application-driven modernization

3. Iterative modernization

Iterative modernization enables organizations to innovate while modernizing (Figure 5). This approach—the one MongoDB recommends—blends data- and application-driven approaches for incremental enhancements, starting with the least complex applications and objects and slowly progressing to more complex ones. With this approach, you can explore iteratively and at your own pace. This one-step-at-a-time approach gives you the best of both worlds: You see immediate gains along the way but are not committing to a newly refactored environment right away. This minimizes risk while preserving data from the legacy systems. The next section explores the iterative approach in more detail.

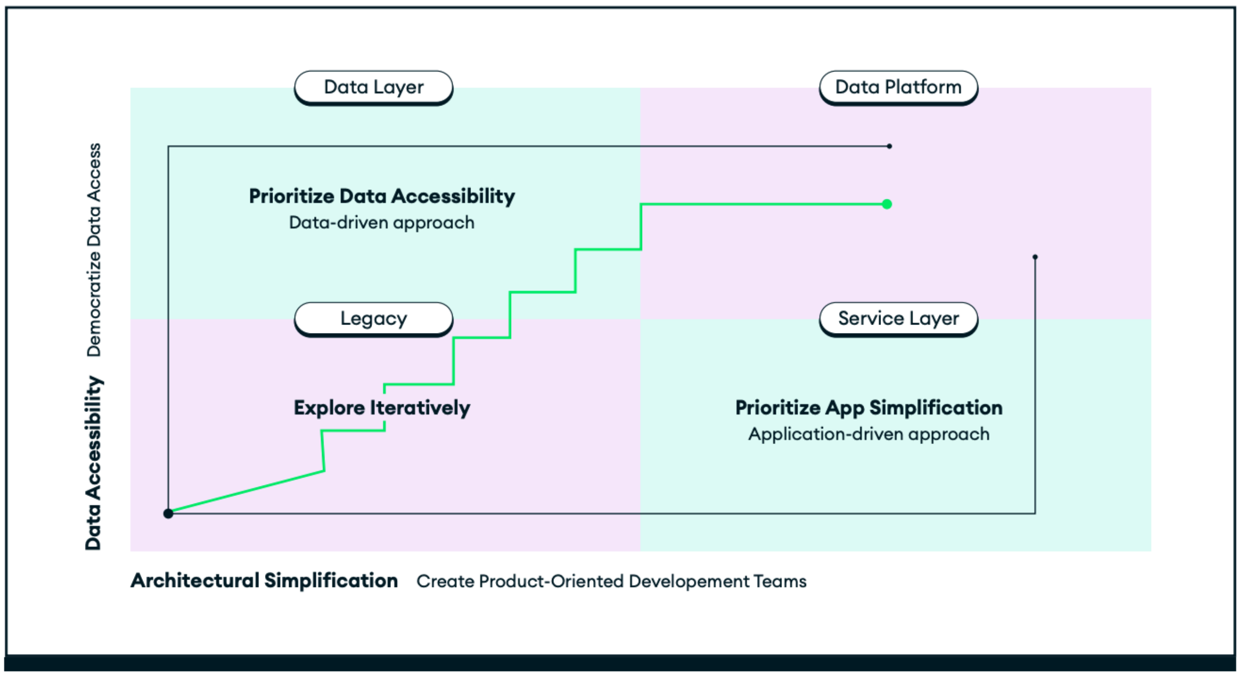
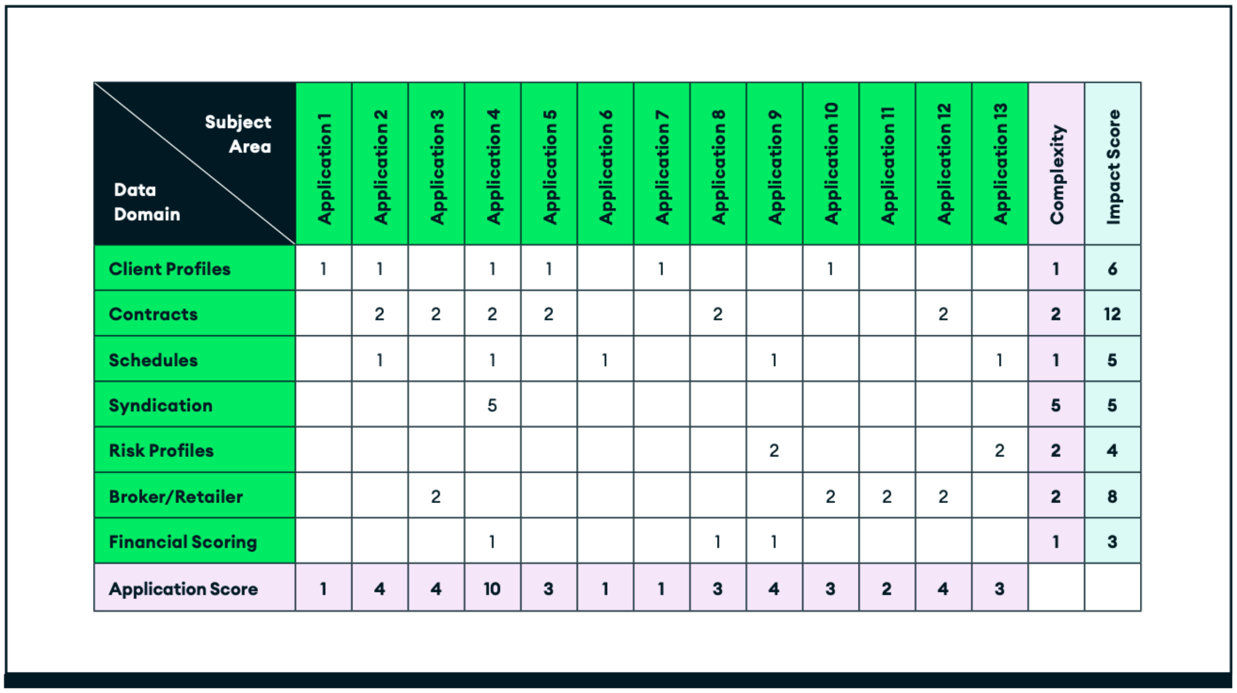

Data Layer
Data Platform
Prioritize Data Accessibility
Data-driven approach
Legacy
Service Layer
Explore Iteratively
Prioritize App Simplification
Application-driven approach
Data Accessibility
Democratize Data Access
Architectural Simplification
Create Product-Oriented Developement Teams

Figure 5: Three approaches to modernization

Modernizing Iteratively

The iterative approach begins by identifying all objects in the application code and any applications that connect to them.

Each of these objects constitutes a data domain (Figure 6). For instance, "client profiles" is a data domain that includes details about clients expressed as values, such as how long they've been a client, their transaction details, and the type of account they have. Once you've identified the objects you're using, you can assign a complexity score to each object based on their properties, methods, collections, and other relevant attributes. You can then identify each application that connects to a domain and rank it based on variables such as how mission-critical it is, how many users rely on it, how many tasks it has to perform, and how complex those tasks are.



Subject Area / Data Domain	Application 1	Application 2	Application 3	Application 4	Application 5	Application 6	Application 7	Application 8	Application 9	Application 10	Application 11	Application 12	Application 13	Complexity	Impact Score
Client Profiles	1	1		1	1		1			1				**1**	**6**
Contracts		2	2	2	2			2				2		**2**	**12**
Schedules		1		1		1			1				1	**1**	**5**
Syndication				5										**5**	**5**
Risk Profiles									2				2	**2**	**4**
Broker/Retailer			2							2	2	2		**2**	**8**
Financial Scoring				1				1	1					**1**	**3**
Application Score	**1**	**4**	**4**	**10**	**3**	**1**	**1**	**3**	**4**	**3**	**2**	**4**	**3**		

Figure 6: Ranking data domains and applications

Ranking the data domains and applications by complexity enables you to create a plan for moving each domain from the legacy system to the new architecture and rerouting applications to connect to the new domains, starting with the least complex data sources and gradually progressing to more complex ones.

In Figure 7, the data domains "client profiles" and "schedules" each have a complexity score of 1 and are used by applications 1, 6, and 7, each with a complexity score of 1. These are perfect candidates to become the first sources of data migrated to the new architecture and the first applications refactored to connect to the new data domains.

Once you have a clear picture of all the objects and applications and have scored them based on the number of dependencies and their complexity, you'll end up with a graph that shows the potential sequence and timing for moving objects and applications into the new data architecture. This will be the basis for your iterative modernization plan.

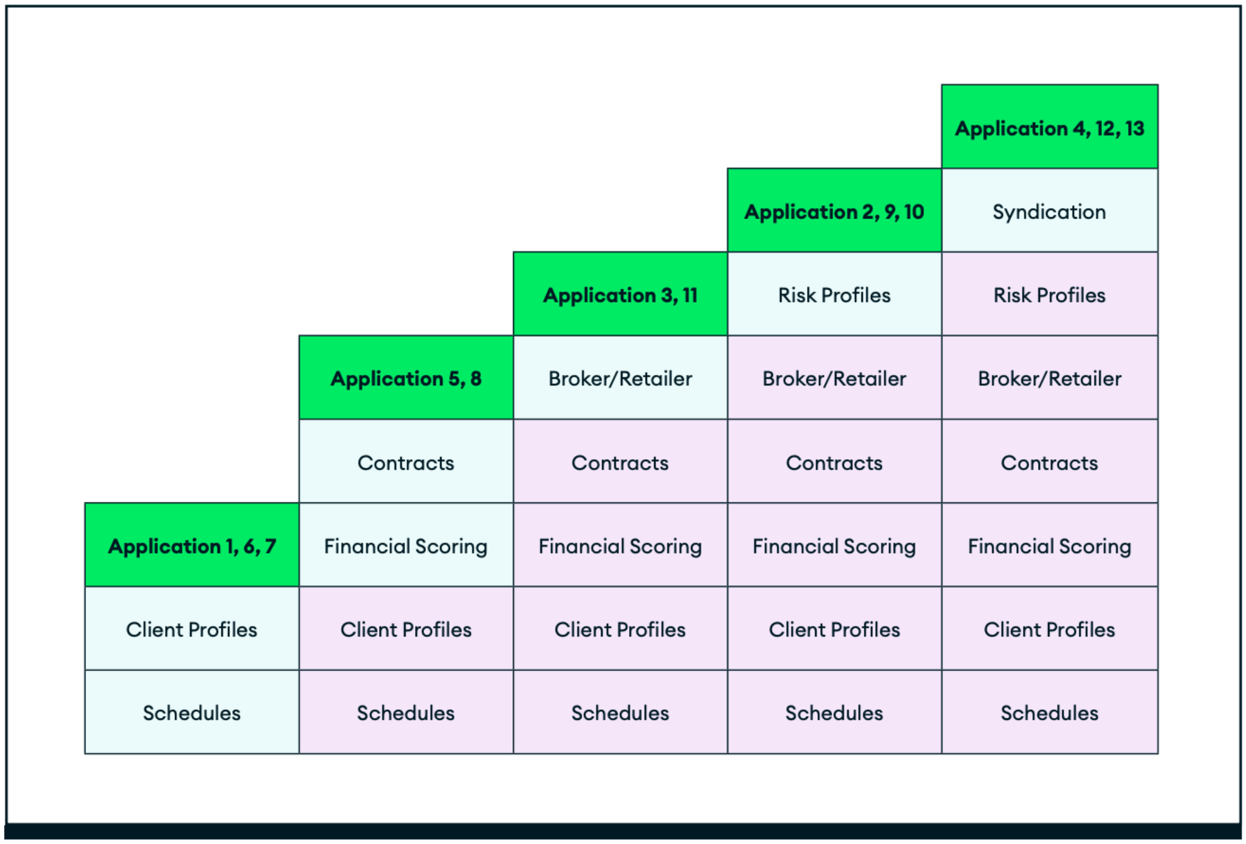


Figure 7: Potential sequence for modernization

Moving Data Between Systems

Before moving any data from the legacy RDBMS to the new environment, you'll need to build temporary scaffolding to transit from the legacy system to the new environment.

The first part of the scaffolding uses connection services between the legacy RDBMS and the new environment. Connection services are needed for three different types of data sources:

1. **Streaming interfaces**. Real-time data, generally measured in seconds, milliseconds, or microseconds, will be replicated between the legacy RDBMS and the new environment simultaneously.
2. **Service interfaces**. End-of-day and batch-processing actions that are common in banking environments.

3. **Specialty connectors**. These connect to specific workloads, like Hadoop or Spark.

Once you've established which connection service you need for each data source, you can begin building the intermediary layer that will bridge the RDBMS to the new data architecture. With the iterative approach, the connection between the legacy RDBMS and the new architecture is the operational data layer (ODL).

Leveraging AI to Accelerate Modernization

In recent years, AI has emerged as a transformative technology with broad applications across various domains, including database migration and application development. Development teams that want to build or utilize a modern operational data layer can benefit from leveraging AI's rapid enhancement for this purpose.

Migrating a legacy application that runs on a relational database for example to a modern database like MongoDB may involve challenges such as schema design, data transformation, code conversion, and testing. The challenges are similar for a team that only has experience with a legacy technology stack trying to develop new applications on a modern operational data layer. Thankfully, AI can significantly streamline these processes, reducing development time, minimizing errors, and improving overall system efficiency. Software technologies are starting to emerge in the market to leverage these advances and here are some examples of how the advancement of AI can help.

Schema transformation and data migration

Transforming an existing relational database schema into another data structure such as a document model to achieve the flexibility benefits can be challenging, especially in a large complex legacy application.

AI can analyze current database schemas, identify data relationships, and suggest optimized target application structures. AI can analyze data patterns within relational tables and recommend document structures, embedded relationships, or references, ensuring efficient data retrieval and storage. By leveraging natural language processing (NLP) or large language models (LLMs), AI can automate schema conversion by identifying fields, data types, and indexes needed for target databases, reducing the manual effort involved.

A significant part of migration involves transforming and transferring data from the existing database to a new modern database. AI can assist in this process by automating data extraction, transformation, and loading (ETL) tasks, ensuring data consistency and integrity throughout the migration.

Code conversion and testing

Migrating to a new modern database often requires significant code changes to accommodate the NoSQL query structure, which differs from traditional SQL. Generative AI (gen AI) can expedite this transition by automating query translation and refactoring application code.

Gen AI, for example, can translate SQL queries for MongoDB’s query API by identifying equivalent operations, adjusting aggregation functions, and generating code that matches MongoDB’s syntax and logic.

AI-powered models can analyze existing code bases and automatically refactor them for new database compatibility, rewriting database interaction code and ensuring that data handling conforms to the new schema and database structure.

Gen AI can create testing scenarios to verify data consistency, query accuracy, and application stability after migration. This ensures that migrated functions are correct and meet predefined performance and reliability standards.

Leveraging AI With MongoDB

AI offers a comprehensive toolkit for developing and migrating applications to a modern operational data layer, addressing challenges across schema design, data transformation, code refactoring, and testing. With its ability to automate and enhance migration tasks, AI empowers organizations to modernize their database infrastructure with reduced risk, lower costs, and improved agility.

When it comes to adopting advanced technologies like ML, gen AI, or AI more broadly, which require data as the foundation, organizations often grapple with the challenge of integrating these innovations into legacy systems. With MongoDB serving as the ODL, the flexible document model enables financial institutions to efficiently handle large volumes of data in real time. By integrating MongoDB with AI, businesses can develop models trained on the most accurate and up-to-date data, thereby addressing the critical need for adaptability and agility in the face of evolving technologies.

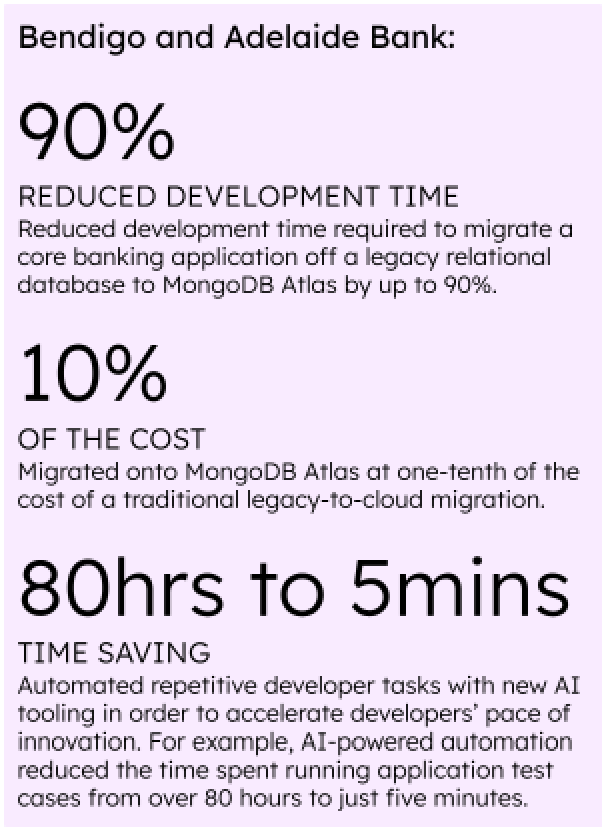
Legacy systems, marked by their inflexibility and resistance to modification, present another challenge in leveraging AI to enhance customer experiences and improve operational efficiency. Challenges also persist, especially in the financial sector, where the uncertainty of evolving AI models over time requires a scalable infrastructure. MongoDB's developer data platform future-proofs businesses with its flexible data schema capable of accommodating any data structure, format, or source. This flexibility facilitates seamless integration with different AI platforms, allowing financial institutions to adapt to changes in the AI landscape without extensive modifications to the infrastructure.

To accelerate the migration journey and adopt a modern data architecture with AI in mind, MongoDB has integrated AI into its services to help customers migrate easier and with greater automation. For example, the MongoDB Relational Migrator uses a combination of rule-based automated intelligence and gen AI to help customers with their data migration and modernization efforts. Relational Migrator can help customers move data snapshots or migrate the entire legacy database from their existing relational system to MongoDB and convert SQL code for MongoDB's query API. Such use of AI can drastically reduce the learning curve and effort to adopt a modern data platform like MongoDB to accelerate the integration and development of modern AI applications.

Bendigo and Adelaide Bank Modernize Core Banking Technology with Gen AI

MongoDB is helping clients modernize their applications by going beyond the database. For example, MongoDB has partnered with Bendigo and Adelaide Bank to modernize its Agent Delivery System (ADS) with MongoDB Atlas as the keystone of an ambitious application modernization initiative. The ADS is a retail teller application for the bank's agent branches used in communities where digital banking functionality is made available from non-bank businesses, like newsagents or pharmacies.

MongoDB empowers all its customers to modernize with applications that are not just future-ready, but future-defining. This is paramount for financial institutions that need to transform quickly and take



Bendigo and Adelaide Bank:

90%

REDUCED DEVELOPMENT TIME

Reduced development time required to migrate a core banking application off a legacy relational database to MongoDB Atlas by up to 90%.

10%

OF THE COST

Migrated onto MongoDB Atlas at one-tenth of the cost of a traditional legacy-to-cloud migration.

80hrs to 5mins

TIME SAVING

Automated repetitive developer tasks with new AI tooling in order to accelerate developers' pace of innovation. For example, AI-powered automation reduced the time spent running application test cases from over 80 hours to just five minutes.

advantage of advancements like generative AI to best serve their customers.

Your Roadmap to Digital Transformation

For years, banks and financial institutions have wrestled with the question of whether to modernize their legacy mainframe systems. With the continued innovation of mobile banking, real-time transactions, analytics, and agile product development, legacy modernization has become a business imperative. Now, with the advent of a seamless, iterative, phased approach to modernization, the case for modernization is as compelling as it's ever been. At MongoDB, we've seen clients save hundreds of thousands of dollars in the first year after modernizing and tens of thousands per month on storage costs. Now, with advancements in LLMs and generative AI, the path to a modernized architecture is faster than ever. And, by taking an iterative approach to modernization, financial institutions can rapidly evolve their tech stacks with very little risk of downtime or disruption to operations.

Learn More

Discover how companies are accelerating their modernization efforts with MongoDB by exploring the following resources

- Innovate with AI: Drive industry success with artificial intelligence and MongoDB Atlas
- The MongoDB Solutions Library is curated with tailored solutions to help developers kick-start their projects
- Accelerate your shift from legacy relational systems to a modern developer data platform with MongoDB's Relational Migrator

About MongoDB

MongoDB's developer data platform offers significant architectural advantages by enabling organizations to securely unify application data (both structured and unstructured) with AI-related data (vectors). This capability allows financial institutions to build rich, real-time AI applications. At the core of MongoDB's developer data platform is MongoDB Atlas, the most versatile multi-cloud database on the market. Atlas provides unmatched data distribution and cloud mobility, built-in automation for resource and workload optimization, and a flexible document model, among other features. MongoDB also offers the flexibility to deploy applications on-premises, on a single public cloud, or across multiple clouds simultaneously, ensuring resilience, scalability, and the highest levels of data privacy and security.

To learn more about MongoDB, visit MongoDB.com

About the authors

Boris Bialek, Vice President and Field CTO of Industry Solutions, leads MongoDB's industry practices, with a focus on the modernization of cross-industry solutions, including financial solutions from core banking, payments and card transactions, trade and risk, and treasury. He is an industry expert in data technologies and a recognized speaker and author. Before joining MongoDB, he worked with FIS, IBM, Dell, and Compaq Computers.
boris.bialek@mongodb.com

Wei You Pan, Director of Financial Services Industry Solutions, leads the financial services industry function within MongoDB. Expertise in financial risk management, loan origination, internet banking and trading systems, he has 20+ years of experience in FSI in various roles.
weiyou.pan@mongodb.com

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.
Case Studies (mongodb.com/solutions/customer-case-studies)
Resources (mongodb.com/resources)
Free Online Training (university.mongodb.com)
Webinars and Events (mongodb.com/events)
Documentation (mongodb.com/docs)
MongoDB Atlas database as a service for MongoDB (mongodb.com/atlas)
MongoDB Enterprise Advanced Download (mongodb.com/try)

Legal Notice