JULY 2021

The 5 Phases of Banking Modernization

Accelerate your digital transformation while minimizing risk

MongoDB.

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, killing innovation and entrenching a fear of failure. They also complicate business requirements that didn't exist when RDBMS were invented, such as data privacy compliance.

Because of the inflexible formats required by table structures, data often becomes duplicated and difficult to analyze.

Legacy modernization is frequently perceived to be time-consuming, complex, and error-prone, but 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.

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'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 best way to modernize is through an iterative model that uses an operational data layer (ODL). The ODL acts as a bridge between a bank's existing systems and its new ones.

This iterative approach 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 migration, 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, 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 data capture system listening to the ODL or a similar system, such as MongoDB Stitch 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.

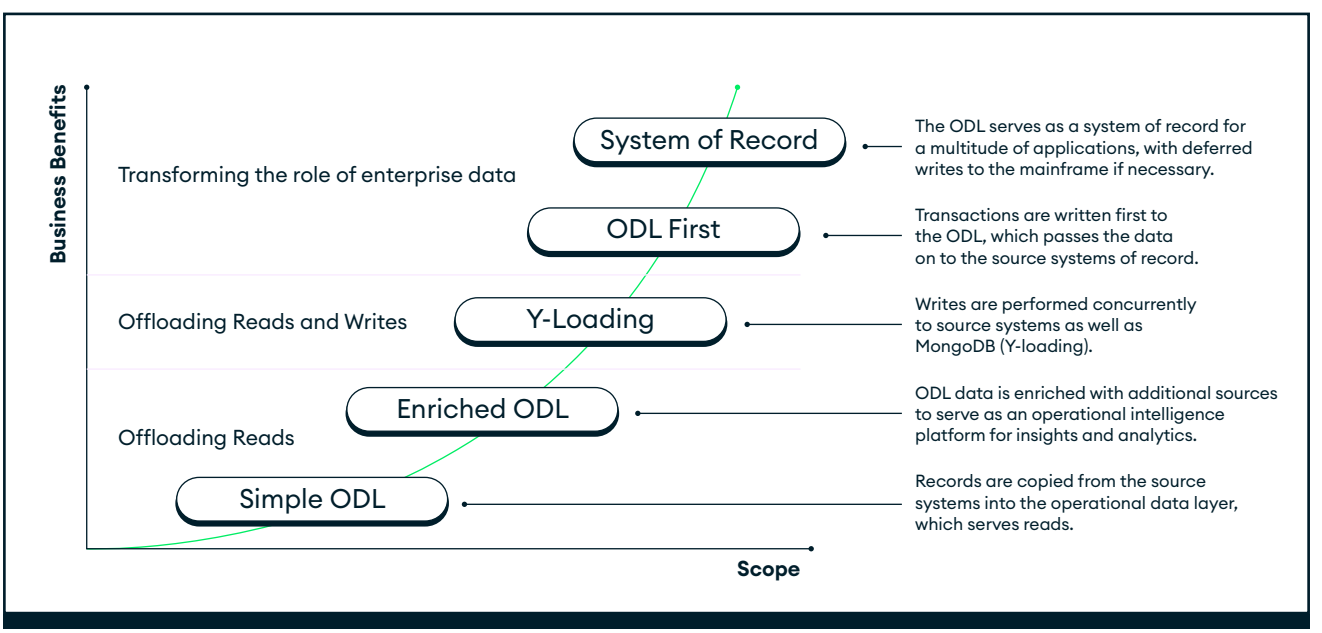

Figure 1: The five phases of banking modernization

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 2). 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.

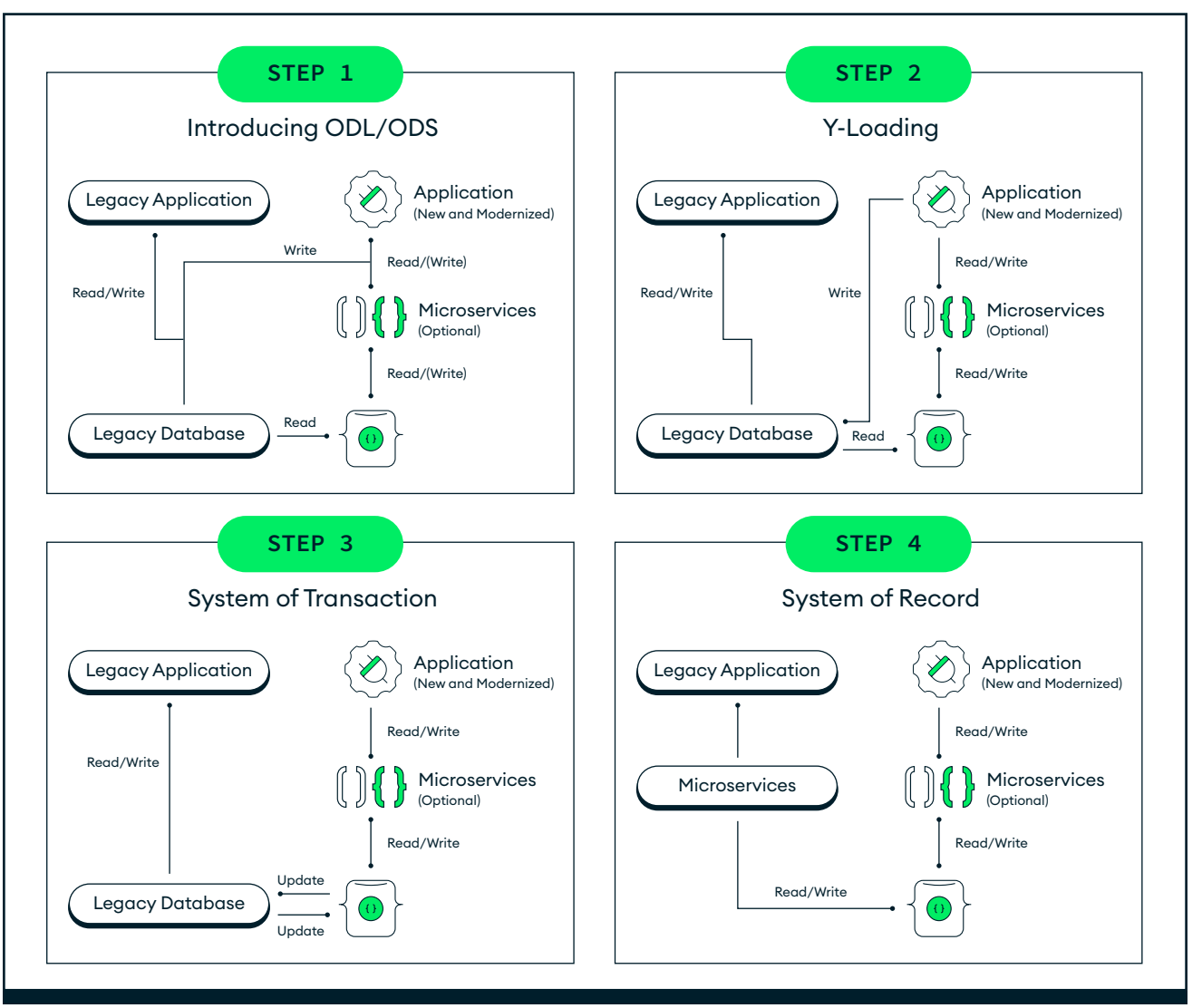

Figure 2: Data-driven modernization

2. Application-driven modernization

With application-driven modernization (Figure 3), 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.

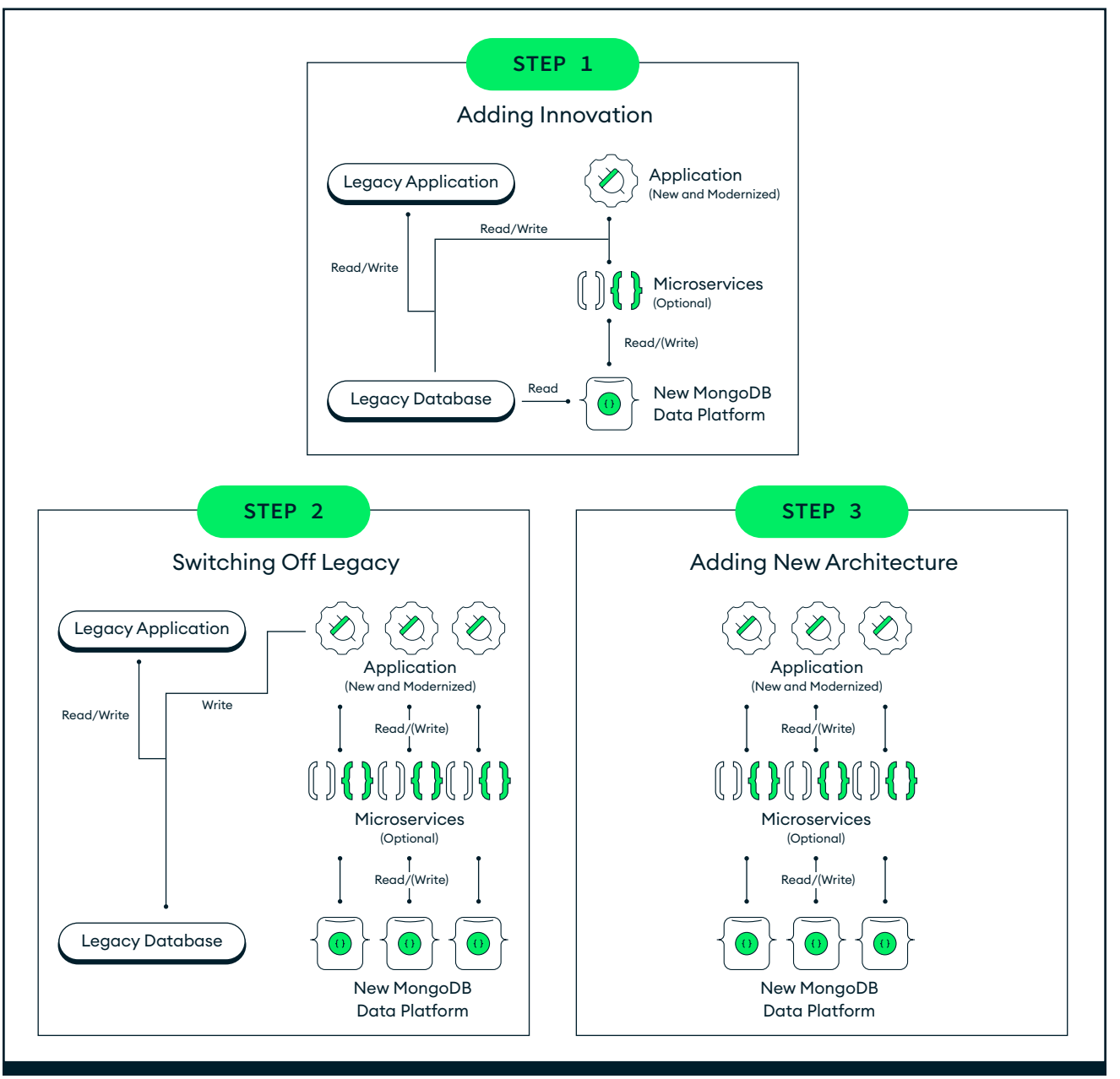

Figure 3: Application-driven modernization

3. Iterative modernization

Iterative modernization enables organizations to innovate while modernizing (Figure 4). 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.

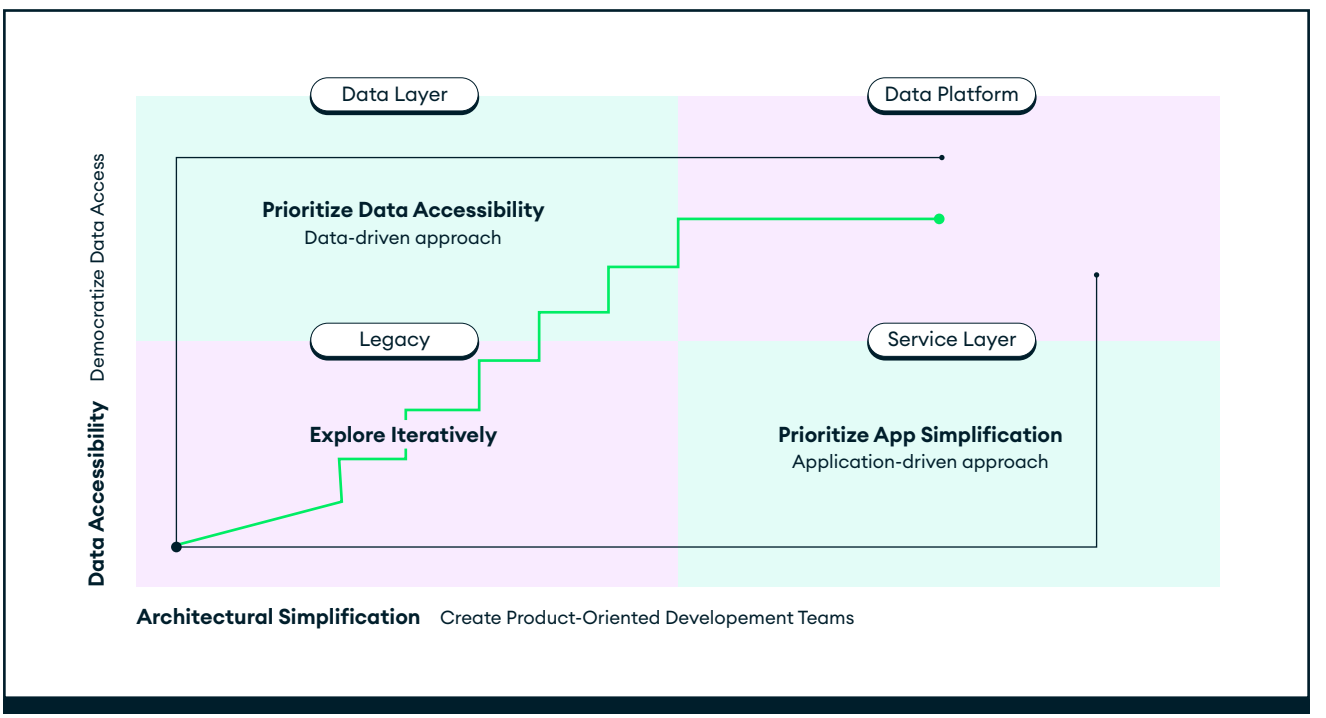

Figure 4: Three approaches to modernization

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, which is a collection of values contained in an element (Figure 5). 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 attributes. You can then identify each application that connects to a domain and rank them 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.

Data Domain \ Subject Area	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 5: Ranking data domains and applications

By ranking the data domains and applications by complexity, you can 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 6, 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 complexities, 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.

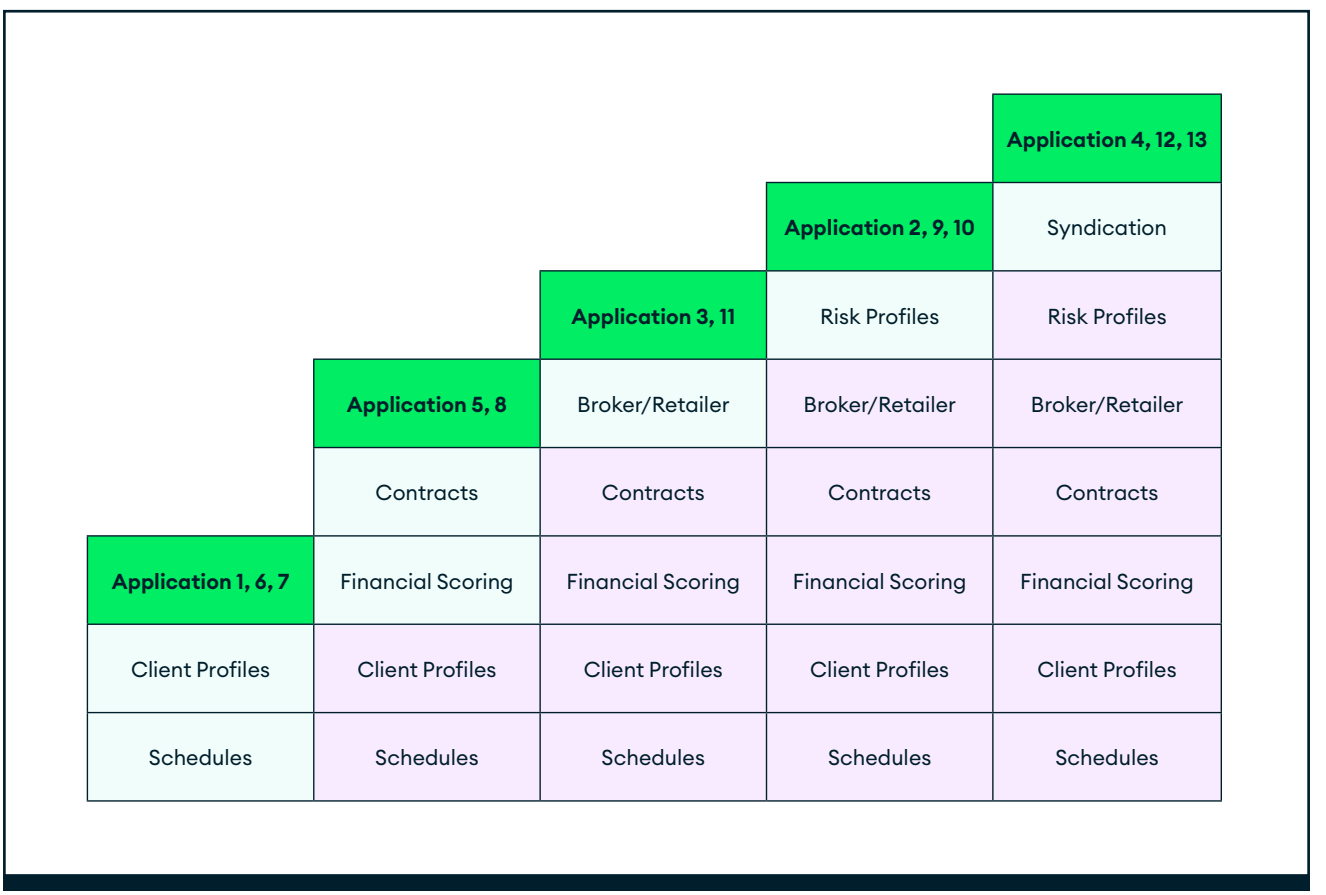

Figure 6: Potential sequence for modernization

Moving Data Between Systems

Before moving any data from the legacy RDBMS to the new environment, you'll need to build temporary scaffolding to transition 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, that 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 modernize-while-innovating approach, the connection between the legacy RDBMS and the new architecture is the operational data layer (ODL).

The Operational Data Layer

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 are provisioned in the new environment.

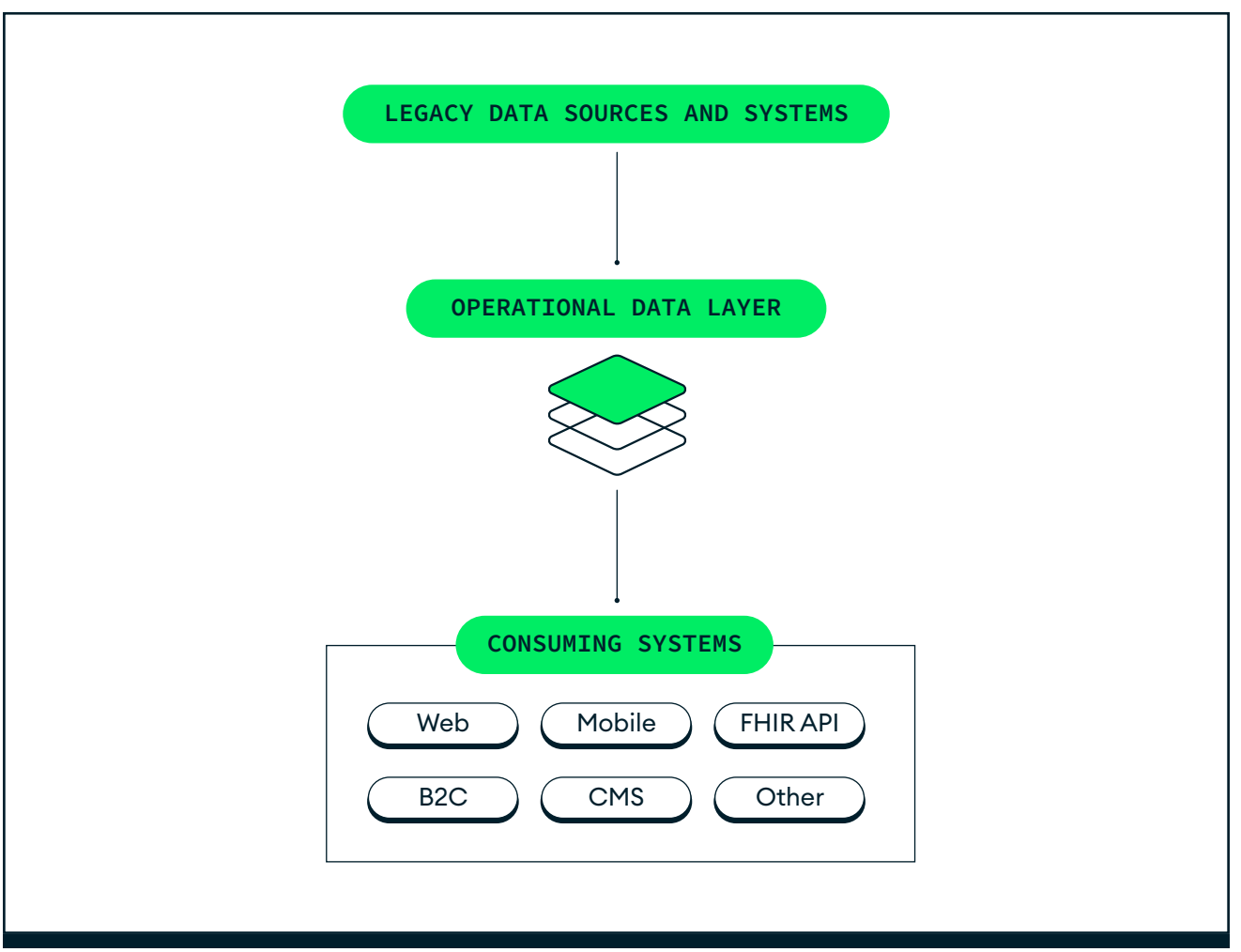

Figure 7: The Operational Data Layer (ODL)

Your Road Map to Digital Transformation

The key to legacy modernization is the bridge between the legacy mainframe and the new architecture, the ODL. It enables banks to offload traffic away from costly mainframe systems and, eventually, to rearchitect monolithic applications into a suite of microservices.

Crucially, by deploying the ODL in a phased approach, banks can embark on their digital transformation journey iteratively, without the risk of an all-or-nothing, rip-and-replace approach. Once the new architecture is in place, development teams can build new business functionality faster and scale new services to millions of users.

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.

For years, banks and financial institutions have wrestled with the question of whether to modernize their legacy mainframe systems. With the emergence 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.

Real-world advantages for financial services firms

45
DAYS
Transaction history expanded from 45 days to 30 months

5
MILLISECONDS
Payment status captured in 5 milliseconds vs. 3 seconds

99.999%
UPTIME
99.999% uptime enabled by five-node replica sets across three data centers

34%
INCREASE
Development efficiency increased by one-third

>10x
STORAGE SAVINGS
Saves tens of thousands a month on storage costs

$10M+
IN COST AVOIDANCE
$10M+ in cost avoidance, including licenses, hardware, and operations

14K+
SERVERS
14,105 servers decommissioned

<400
MILLISECONDS
Response time latency <400 milliseconds for queries deep in transaction history

Learn More

Discover how companies are accelerating their modernization efforts with MongoDB.

About the Author

Boris Bialek, Global Head of Industry Solutions, leads MongoDB's industry practices, with a focus on the modernization of finance solutions, including 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 for many years with FIS, IBM, Dell, and Compaq Computers. He has a Master of Science degree from the Karlsruhe Institute of Technology.
boris.bialek@mongodb.com