Architecting High Performance Financial Ledgers

Patterns and Practices for Modern Applications

November 2024

Written by: ***Kris Hansen***
CTO, **Sagard**

Sagard

Table of contents

- Executive Summary
- Introduction
- Understanding Ledgers in Financial Services
- Advancements and Trust in Ledger Systems
  - The Evolution of Ledger Technology
  - Building Trust in Ledger Systems
  - Reconciliation Processes
  - Ledger Integration and Interoperability
- Key Features and Architecture of Modern Ledgers
  - Key Components of a Modern Ledger
  - The Ideal Ledger Architecture
- Technical Considerations for High-Performance Ledgers
- Case Studies and Real-World Examples
  - Successful Implementations
  - Problematic Designs
  - Industry-Specific Adaptations
  - Case study: Formance ledger architecture
- Build vs. Buy Considerations
- Emerging Trends and Future Directions
  - Blockchain Integration
  - AI-Assisted Reconciliation and Anomaly Detection
  - Real-Time Reporting and Analytics
  - Distributed Ledger Technologies
  - Regulatory Technology (RegTech) Integration
- Regulatory Compliance and Ledger Design
- High Performance Reference Architecture
  - High Performance Transaction Layer
  - Accounting Layer
  - Durability and Workflow Layer
- Conclusion

Executive Summary

There are some challenging aspects of building financial services businesses including banks or fintechs - one of these is how to design a ledger which can scale and perform. Customers demand precision at scale and regulators insist on accuracy and clarity around who is holding which funds and where. Building a great ledger may not sound very exciting but it is a foundational part of getting a financial services business right and doing this poorly can be catastrophic.

The concepts around booking debits and credits have not changed materially since they were introduced but technology has evolved rapidly - and continues to evolve. It is now possible to develop more scalable and consistent ledgers than ever before. This document outlines some of the core principles to consider when acquiring or building a ledger and provides some examples of ledger components and capabilities to evaluate when making these decisions. Buying legacy technology or building with legacy approaches would leave a financial services firm with a significant disadvantage when it comes to performance, efficiency and regulatory readiness.

A practical and more detailed example is provided to help make some of the core concepts around capabilities and performance real - third party off the shelf components provide transparent examples of how to manage the more difficult aspects of ledger systems design and implementation. These are intended as points of reference and also can help technology teams connect the world of systems architecture to reality in the form of real applications and code.

Introduction

I've spent the last two decades creating, building and integrating with financial services ledgers. From designing and purchasing to building these systems for both traditional banks and leading-edge fintechs, I've seen a lot of interesting things in this space. In fact, I've had the chance to review the technology architecture of about 400 fintech firms and a dozen banks across the globe. I've seen the good, the bad, and the ugly when it comes to ledger systems.

This document isn't just another theoretical piece. It's a distillation of real-world experiences, battle-tested insights, and hard-earned lessons. **My goal? To guide you through the maze of creating ledger systems that are not just functional, but robust, scalable, and flexible enough to handle whatever the fintech world throws at them.**

You are going to have bad days in financial services - partner systems go down, networks have outages, payments get reversed, settlement files go missing, trades come in duplicated and don't get me started on the long tail of card authorization flows and reversals. Payments is a messy business and your ledger should be there to bring order to the chaos so that when you have these bad days you can make the correlations and adjustments to make things right. Your ledger is often your final layer of clarity and truth - or should be if you get it right.

The time to start thinking about how your ledger will scale both technically and functionally is not when it is under load, it's during the design, planning and development phases. The mantra in most startups is to move fast and break things and while this is generally true it doesn't really work for core financial information as well as regulatory hard lines. I'd suggest that moving fast, anticipating things is a better approach in fintech.

When I'm assessing a fintech solution, one of my first actions is to take a good, hard look at their ledger design. It's like a window into how the team really thinks about the "fin" side of fintech. So, consider this document a roadmap to designing, provisioning, and implementing top-notch ledgers. Let's dive in.

Kris Hansen
Chief Technical Officer,
Sagard

Understanding Ledgers in Financial Services

Ledgers are at the core of any financial services business.

Traditionally in financial services a 'core banking system' really refers to the core subledgers for deposits, loans and other financial products. The subledger is really the core source of truth for the things that most financial institutions are most concerned with from a financial, operational and wellbeing perspective. If you have a financial services business but you don't know where the money is, this is a problem - so having a solid ledger or core is more than just a nice to have. This is the essence of the business.

The fundamental purpose of a ledger is to track the location and movement of funds. This seemingly simple task is critical to the functioning of any financial institution. Without accurate and reliable ledgers, the entire financial system would be built on an unstable foundation.

I've witnessed firsthand how a well-designed ledger can be the difference between a thriving financial institution and one that struggles with basic operations. It's not just about keeping the books balanced, although that's crucial. It's about providing the bedrock of trust and accuracy that every financial transaction relies on.

As we delve deeper into ledger architecture, it becomes clear that this often overlooked aspect of finance is actually a complex and essential component that demands careful consideration and robust design.

If your ledger is a simple table with a list of transactions - this is more of a transaction log and when faced with edge case scenarios and reconciliation challenges you will find out **why banks over the years have built out so much robustness around their subledger designs – it's not for the happy path of transactions, it's to handle all of the many corner cases**.

> "It's not just about keeping the books balanced, although that's crucial. It's about providing the bedrock of **trust** and **accuracy** that every financial transaction relies on."

Understanding Ledgers in Financial Services (cont.)

Ledger Types and Their Roles

It's crucial to recognize that ledgers are not one-size-fits-all solutions. In my work across various financial institutions, I've encountered ***two main types of ledgers****, each serving distinct purposes:*

1. **Enterprise General Ledger (EGL)**: This type of ledger bears the additional burden of financial and regulatory reporting. It needs to handle complex requirements like Basel II / Basel III compliance, IFRS treatment of different asset types, and various regional reporting topics. For these purposes, I have typically seen (and recommended) the use of off-the-shelf software with built-in reports and calculations.

2. **Subledgers**: These focus on specific product areas or domains and would roll up (where relevant) to an enterprise GL. This is typically the layer of focus for a fintech wanting to build its platform with a solid base, and it's where much of my work has been concentrated and is the focus of this document.

For fintech platforms, the focus is often on building (or considering buying) more of a subledger. This approach allows for greater flexibility and customization in handling specific product or domain requirements, while still providing the necessary data to feed into an enterprise GL for broader financial reporting.

When is an EGL needed vs. a subledger? Generally speaking when the complexity of the business demands it: this usually means multiple business lines, multiple accounting treatments for different geographies (say IFRS as well as US GAAP), consolidating product lines. Where a subledger is the source of truth for the specific balance of a specific account the EGL is the source of truth for the overall position. Subledgers should be ready to 'roll up' into an EGL but should be careful not to duplicate the capabilities of an EGL.

In my experience, understanding the distinction between these ledger types and their roles is crucial for designing an effective overall financial architecture. It helps in making informed decisions about where to invest resources and how to structure your ledger system to best serve your specific business needs.

Advancements and Trust in Ledger Systems

The Evolution of Ledger Technology

The landscape of ledger technology has evolved significantly over the years. From a pure technology standpoint, it's easier than ever to scale for performance and flexibility. However, despite these advancements, I continue to encounter many poorly thought-out ledger designs that lack essential accounting concepts and durability.

There has been significant growth in the world of database design over the last two decades - distributed file systems, NoSQL databases, Big Data Processing Frameworks, real time event sourcing, in memory data processing and data warehousing innovations. Yet most of these advancements have not directly impacted the design of ledgers, this I believe is largely due to the challenge of needing to provide a unary answer to questions around accounts and amounts. If you develop a ledger with a single table providing the answers around which accounts have which balances and then you cluster it now you have two problems: locking and keeping the data in sync across multiple potentially locked rows on different instances.

This disconnect between technological capability and practical implementation is a recurring theme I've observed across numerous fintech firms and banks. It's a stark reminder that having access to advanced technology doesn't automatically translate into an effective ledger system.

One of the hard truths I've learned is that **ledgers can fail on edge case scenarios, and when they do, it can lead to severe consequences**. I've seen instances where a seemingly minor oversight in ledger design has resulted in major financial discrepancies and regulatory issues. These failures can be both technical and functional in nature. Having a ledger which fails under occasional peak load is bad but having a ledger which is subtly off with interest calculations and posting and to have something like this go unnoticed for a few quarters would be in many ways worse.

The challenge, therefore, lies not just in leveraging modern technology, but in doing so while maintaining the fundamental principles of accounting and ensuring the system's resilience in all scenarios. It's a delicate balance that requires both technical expertise and a deep understanding of financial processes.

Advancements and Trust in Ledger Systems (cont.)

Building Trust in Ledger Systems

Trust is paramount in financial systems, and ledgers are no exception.

Throughout my career, I've identified several key factors that contribute to building and maintaining trust in ledger systems:

1. **Comprehensive referential information**: Carrying and maintaining as much referential information as possible is crucial. This data helps with traceability and provides context for each transaction.

2. **Double-entry bookkeeping structures**: The ability to balance accounts using double-entry bookkeeping principles is not just an accounting requirement—it's a fundamental aspect of building trust in your ledger system.

3. **Granular account structuring**: By structuring accounts in as fine-grained a way as possible, it becomes feasible to show the micro-flows of funds and how these aggregate into larger transactions. This level of detail can be invaluable for auditing and troubleshooting.

4. **Detailed fund flow steps**: Maintaining detailed steps associated with fund inflows and outflows can help identify small issues which, in the world of ledgers, can sum up to major problems. I've seen cases where overlooking minor discrepancies led to significant financial implications down the line.

In my experience, these elements work together to create a ledger system that is not just accurate, but also transparent and auditable. This level of trust is essential, particularly in today's regulatory environment where scrutiny of financial systems is at an all-time high.

Advancements and Trust in Ledger Systems (cont.)

Reconciliation Processes

Another critical aspect of maintaining trust in ledger systems relates to reconciliation processes. A robust ledger is invariably accompanied by a solid set of processes to reconcile accounts, especially against any internal and external counterparties.

In my work with various financial institutions, I've found that effective reconciliation processes typically include:

1. **Regular internal reconciliations**: This involves cross-checking different internal systems and accounts to ensure consistency across the organization.

2. **External counterparty reconciliations**: Regular checks against external parties (such as banks, payment processors, or other financial institutions) are crucial to identify and resolve any discrepancies quickly.

3. **Automated reconciliation tools**: While manual reconciliation is sometimes necessary, automated tools can significantly improve efficiency and reduce human error.

4. **Clear escalation procedures**: When discrepancies are found, having a clear process for investigation and resolution is crucial.

Reconciliation would be easy if the data were perfect – this is of course never the case. Payments data is about as messy as data can get and reconciliation processes see the worst of the worst. Card transactions are notoriously painful to reconcile given the convoluted nature of the authorization settlement process. Part of the subledger's job is to help resolve the mysteries of suspense and identify payment flows which don't get resolved. Reconciliation is the key to finding defects in systems and processes which can allow for the leakage of funds or the printing of funds, both of which are very bad scenarios.

These processes ensure the ongoing accuracy and reliability of the ledger system. They act as a safety net, catching errors or inconsistencies before they can snowball into larger issues. In my experience, **robust reconciliation processes are often the unsung heroes of financial operations, quietly maintaining the integrity of the entire system**.

Advancements and Trust in Ledger Systems (cont.)

Ledger Integration and Interoperability

Integrating ledgers is a well known pain point in the industry.

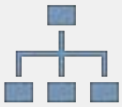
When integrating ledgers there are a few core principles which can help make this a bit more manageable:

1. **Have a clear understanding of which ledger is primary and which is shadow and for which objects** - one must be the lead ledger for key financial information. Mixing this up creates complexity and chaos.

2. **Understand the data structures and which will be truncated where and when (and how) so that key information is not lost in the integration.** This is common with metadata which may not seem essential for the core booking of a transaction but may be critical in the understanding of the transaction context for regulatory purposes.

3. **Overlap identifiers:** Two ledgers which are integrating should carry or have mapped the unique identifiers for the objects that they are sharing or interacting with. This can make reconciliation easier and can help rebuild a common view of events in the case of a transactional disaster such as overwritten or missing records.

Key Features and Architecture of Modern Ledgers



Key Components of a Modern Ledger

To truly understand ledger systems, it's crucial to recognize that a ledger is not a monolithic entity, but rather a system comprised of several key capabilities.

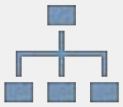
Based on my experience, here are the essential components:

> A ledger is not a monolithic entity, but rather a system comprised of several key capabilities."

1. **High-performance transaction handling**: This component needs to manage the high-volume deluge of real-time transactions, such as trades or card transactions, without compromising speed or accuracy.
2. **Accounting treatment and internal account assignments**: This ensures adherence to double-entry bookkeeping principles, a fundamental aspect of financial record-keeping that's critical for maintaining the integrity of your financial data.
3. **Flexible transaction metadata**: This allows for rich contextual information to be associated with each transaction, providing deeper insights and facilitating more detailed analysis.
4. **Immutable record storage with closed periods**: Once a transaction is recorded and a period is closed, it should remain unchanged. This creates a reliable audit trail and builds trust in the system.

Each of these components plays a vital role in creating a comprehensive and effective ledger system. In my years of reviewing and designing ledger systems, I've seen how neglecting even one of these elements can lead to significant issues down the line.

Key Features and Architecture of Modern Ledgers (cont.)



The Ideal Ledger Architecture

An ideal ledger architecture, in my experience, strikes a balance between performance and flexibility.

It typically includes two key components:

1. A highly transactional component focused on accounts, amounts, and balances. This serves as the core engine, handling the day-to-day financial operations with speed and accuracy.

2. A robust data structure for flexible metadata. This allows for rich, contextual information to be associated with transactions, providing deeper insights and adaptability.

The key to a successful ledger lies in the method of holding these two data structures together in a durable and performant fashion. This dual-structure approach allows for both the high-speed processing of financial transactions and the flexible management of associated data, creating a system that is both powerful and adaptable.

In my experience in designing and reviewing ledger systems, I've seen many that are advanced in one aspect but fall short in the other. The most effective systems manage to integrate these components seamlessly, providing a solid foundation for financial operations while remaining flexible enough to adapt to changing business needs.

Technical Considerations for High-Performance Ledgers

On the technology side, supporting high transaction volumes safely and securely is a key requirement for modern ledgers.

Based on my experience designing and implementing high-performance ledgers, here are some critical technical considerations:

1. **Account locking and blocking mechanisms**: A ledger under pressure needs to be able to safely block and lock accounts in a way that does not degrade performance. This is crucial for maintaining data integrity in high-volume environments. A ledger architecture can only perform as fast and as safely as its slowest locking mechanism.

2. **FIFO transaction sequencing**: Ensuring transactions are kept in a First-In-First-Out sequence is vital for accurate record-keeping and reporting.

3. **Scalability planning**: It's crucial to remember that whatever transaction volume you think you need, the actual number is probably higher and always growing as you scale. I've seen many systems struggle because they weren't designed with sufficient headroom for growth. With ledgers there are often unary databases or structures (often tied to the locking mechanism) which are simply impossible to parallelize.

4. **Partner system failure handling**: Consider scenarios where a partner system (like a payment switch) goes down. When it comes back up, it will likely attempt to fire as many transactions as possible. Your system needs to be ready for this scenario to avoid being overwhelmed.

5. **Recovery mechanisms**: Robust recovery processes are essential for handling system failures or data inconsistencies. This includes both technical recovery (like database rollbacks) and business process recovery (like reconciliation and correction procedures).

In my years of working with ledger systems, I've found that these technical considerations often make the difference between a system that can handle real-world pressures and one that falters under strain. Addressing these points early in the design process can save significant headaches down the line.

Case Studies and Real-World Examples

I've encountered a wide range of ledger implementations, both successful and problematic. These real-world experiences offer insights into effective ledger design and common pitfalls to avoid.

Successful Implementations

One standout example I encountered was a mid-sized fintech that built a highly flexible ledger system. Their key to success was **a modular design that allowed for easy addition of new financial products**. This flexibility enabled them to rapidly enter new markets and adapt to changing regulations without overhauling their entire system.

Another notable case was a traditional bank that successfully modernized its legacy ledger. They achieved this **by implementing a staged migration strategy, gradually moving from their monolithic system (used for transactional data) to a more distributed architecture augmented with flexible transactional metadata.** This approach minimized disruption to ongoing operations while significantly improving performance and scalability.

Problematic Designs

On the flip side, I've also seen my fair share of ledger designs that fell short. One particularly memorable case involved a startup that prioritized speed over accuracy in their initial design. While their system could handle high transaction volumes, it lacked robust reconciliation processes. This led to significant discrepancies that were only discovered months later, resulting in a costly and time-consuming cleanup operation.

Performance problems are hard to see until they are experienced. Performance testing without realistic data (which is difficult to simulate) often only shows the 'happy path' of ledger flows and having a ledger which takes many seconds or minutes to return a balance or commit a transaction is a critical failure for most financial systems and one that is very difficult to recover from because the solution is often a mix of business functional changes as well as technical remediation. **Restructuring accounts, changing the way that transactions are being booked and adding more systems capacity** being an example of the complex solutions that may be required in a situation like this. **It is far preferable to anticipate and design with margins of error vs having to course correct on the fly.**

Case Studies and Real-World Examples (cont.)

Another common issue I've observed is the **underestimation of regulatory requirements.** Several firms I've worked with initially designed their ledgers without full consideration of compliance needs, necessitating extensive and disruptive retrofitting later on.

Industry-Specific Adaptations

Different industries often require unique adaptations in their ledger architectures. For instance, in the insurance sector, I've seen ledger systems designed to handle complex, long-term contracts with multiple payment schedules. In contrast, ledgers in the retail banking space often prioritize high-volume, real-time transaction processing.

One interesting example is the *Formance* architecture, which I've seen successfully implemented in several fintech startups. Its **event-sourcing approach provides a high degree of flexibility and traceability**, which has proven particularly valuable in rapidly evolving regulatory environments.

These case studies underscore the importance of tailoring ledger architecture to specific business needs while adhering to fundamental principles of accuracy, scalability, and compliance.

Case study: *Formance* ledger architecture

As an example of a ledger which is well designed and open source (so easy to analyze compared to say, a bank's custom ledger) let's take a look at the Formance *ledger.* Formance *has designed a ledger platform that is implemented in Go and uses PostgreSQL for data storage.*

They have developed this project well beyond a simple table with accounts and amounts and have developed the following key capabilities:

- **Core Functionality**: Implements a ledger system for financial operations, using a command-based architecture for transactions, reversals, and metadata management.
- **Data Model**: Deals with transactions, accounts, balances, and logs, supporting expanded views for additional context.

Case Studies and Real-World Examples (cont.)

- **Storage**: Uses a **ledgerstore** package for data persistence, with querying support for various entities.
- **Command Pattern**: Employs a **Commander** struct for business logic and consistency, with a command compiler for interpreting operations.
- **Concurrency and Locking**: Uses a locking mechanism to ensure data consistency in concurrent operations.
  Locking is a key aspect of ledger design - but every locking strategy has a theoretical and practical transaction per second limit. Ledgers are hard to federate reliably so most will have a single locking FIFO queue and many will use the database to handle locking by using row level locking. For this ledger design they are using a custom application level locking mechanism which locks at the account level and distinguishes between read and write operations. When a transaction is initiated, the ledger will acquire account level locks for each account involved in the transaction and if one of the account is unavailable the lock request is queued and the application will poll to check when the lock is released and the release process also triggers a recheck of the queued lock intents. Overall this is well thought out, the chained locking might lead to a lower transaction per second throughput but this is helped by more granular locking logic.
- **Event Publishing**: Integrates with a message publisher for event-driven architecture, using a **LedgerMonitor** for tracking events.
  Events and ledgers go hand in hand - many other services want to subscribe to ledger events.
- **Metadata Management**: Supports CRUD operations for metadata on various entities.
  Separating the treatment of core transactional data (accounts and amounts) from transactional metadata (can be anything) is a key aspect of modern ledger design which allows for high performance on the transactional side and flexibility on the metadata side.
- **Error Handling**: Implements custom error types for management and reporting.
- **Context-based Operations**: Most methods use context.Context for cancellation and value propagation.

Case Studies and Real-World Examples (cont.)

- **Cursor-based Pagination**: Uses a generic Cursor type for efficient handling of large datasets.
- **Aggregation**: Supports aggregated balance queries for summarized financial views.
- **Schema Management**: Includes functionality to check if the database schema is up to date and migrations using a custom migration package which executes all migrations within database transactions.
  Migrating a live running ledger is a terrifying experience, having robust tooling and well controlled package for this is a good approach. Although there are many standard Go packages for migration management using their own here ensures control of a critical aspect of ledger management.
- **Scripting**: Supports running scripts for creating transactions, allowing flexible, complex operations.
- **Modularity**: Well-structured with clear separation of concerns between different packages.
- **Observability**: Integrates telemetry logging for debugging and monitoring.

As you can see from the list of capabilities above, a well designed ledger is more than a single table with debits and credits. Not all projects will need all of this functionality but core features like locking, metadata management, schema management and having a well thought out data model are critical and not easily implemented.

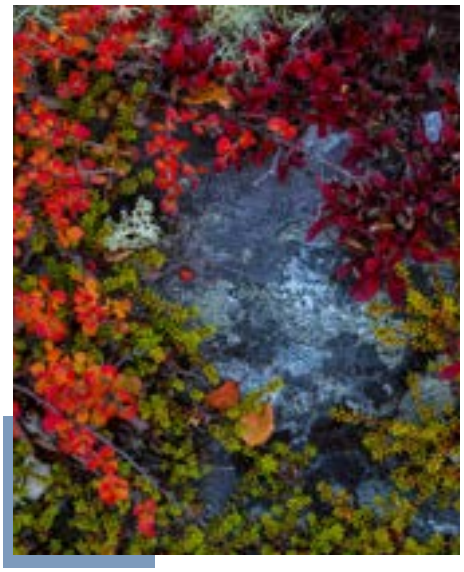
Build vs. Buy Considerations

When it comes to ledger systems, one of the most critical decisions is whether to build a custom solution or use an existing package, framework or open source project.

In some cases, building a ledger might be essential for your business. This is particularly true if you have unique requirements that known solutions can't meet, or if ledger functionality is a core differentiator for your product. Building your own system provides maximum control and customization potential.

However, it's worth considering whether this core functionality could be offloaded to a team or project solely focused on ledger systems. These specialized firms often have deep expertise and economies of scale that can be hard to match in-house. They may also offer features and regular updates to keep pace with regulatory changes.

Also when you consider this question, keep in mind that there is a difference between a **subledger** (for say a cards processing system) and an **enterprise general ledger** - largely intended for consolidation and regulatory reporting. I would be much more likely to advise the build of a subledger vs an enterprise GL. The subledger is going to be much closer to the end product behavior and the enterprise GL will be much more exposed to regulatory changes.



The other consideration in this decision is your engineering culture. Buying a ledger as a subledger which needs to be deeply integrated into product functionality which is being built by an in-house team will affect product velocity and will add complexity. Needing to rely on external systems and potentially consultants to design and implement new product features hampers innovation and product agility. All of these factors can adversely affect the product / engineering culture. Having a purchased ledger product which can be operated and managed by the product/ engineering team can be the best of both worlds: not having to build it all but being able to maintain the optimal product / engineering velocity.

Build vs. Buy Considerations (cont.)

> Don’t invent anything you don’t need to - because you may find things which you really need to invent and your capacity will be needed for this.”

In my experience, the decision often comes down to a careful cost-benefit analysis. While building allows for precise customization, it also requires significant time, resources, and ongoing maintenance. Adopting an existing package or solution, on the other hand, can offer a faster time-to-market and reduced maintenance burden, but may require compromises on specific functionalities.

My general advice in this space is don’t invent anything you don’t need to - because you may find things which you really need to invent and your capacity will be needed for this. And if you do need to build your ledger try to lean on patterns and methods that are proven and only deviate as needed. I have seen many teams not only building their ledger but also attempting to reinvent aspects of financial accounting which have been practiced since the first ledger was etched into a stone tablet. A nice mix of proven development patterns and frameworks and knowledge of generally accepted accounting principles and practices makes for a solid ledger team.

Emerging Trends and Future Directions

The landscape of ledger technology is continually evolving, and staying ahead of these trends is crucial for building future-proof systems. Based on my observations and interactions with industry leaders, here are some key developments to watch:

Blockchain Integration

While the initial hype around blockchain has settled, I'm seeing increased interest in integrating blockchain technologies with traditional ledger systems. This hybrid approach aims to combine the transparency and immutability of blockchain with the speed and regulatory compliance of conventional ledgers. This is happening in Web3 companies but also in more traditional firms. Blockchain approaches are strong for distribution and sharing ledger visibility (supporting trust) they are not strong in high transaction throughput and latency; pairing this technology with something highly transactional can achieve results.

AI-Assisted Reconciliation and Anomaly Detection

Many of the challenges associated with reconciliation, settlement, and dealing with suspense comes from imperfect data in the payments and banking industry (and I'm being generous here) . Reducing the manual activity required to make sense of this data is a great use case for AI/ML. I've seen AI used for merchant categorization of card payments data and bank fee analysis. Recently, I've also seen AI models successfully identifying fraud and money laundering patterns and inconsistencies that would be challenging for human auditors to spot, especially in high-volume environments.

Real-Time Reporting and Analytics

The demand for real-time financial insights is driving innovations in reporting and analytics capabilities. Modern ledger systems are now expected to provide instant visibility into financial positions, often through intuitive dashboards and self-service analytics tools.

Emerging Trends and Future Directions (cont.)

Distributed Ledger Technologies

Beyond blockchain, other forms of distributed ledger technologies are gaining traction. These systems promise to improve inter-organizational reconciliation and create more efficient, transparent financial ecosystems. However, their adoption is still in early stages, and I anticipate significant developments in this area over the coming years.

Regulatory Technology (RegTech) Integration

As regulatory requirements continue to evolve, I'm seeing increased focus on integrating regulatory technology directly into ledger systems. This trend aims to automate compliance processes and reduce the risk of regulatory breaches.

While these trends offer exciting possibilities, it's important to approach them with a balanced perspective. In my experience, successful adoption of new technologies in ledger systems requires careful evaluation of their practical benefits and potential risks.

Regulatory Compliance and Ledger Design

There are many regulations which care about who is moving money and where they are moving it to - and where the money exists. There are even some regulations which speak directly to the ledger and where and how ledger processing takes place. It's important to consider the regulatory stakeholders and consider the first principles in the ledger design.

Having an audit forward design involves holding as many internal and external identifier correlations on or near the ledger as possible. Ideally the ledger can serve as a lingua franca for all of the systems and counterparties it interacts with. I consider this as a key requirement for the payment metadata layer of the ledger: when receiving a payment catch the unique identifier from the sending system and when sending a payment to a third party capture the transaction identifier from the recipient. This seems duplicative and will often not be required but when it is required it means that something has gone very wrong and these identifiers can help trace and unravel ledger related mysteries.

Much of regulatory compliance can be resolved with clarity of data and transparency in reporting. A ledger which carries this data and then sourcing that data into a data warehouse so that the same data can produce different (but similar) reports for different regulatory stakeholders is a pattern that I have used and seen used successfully by most in the financial services industry. The challenge here is to make sure that the data warehouse does not become the ledger and provide clear prescriptive guidance on where the ledger and warehouse data begins and ends.

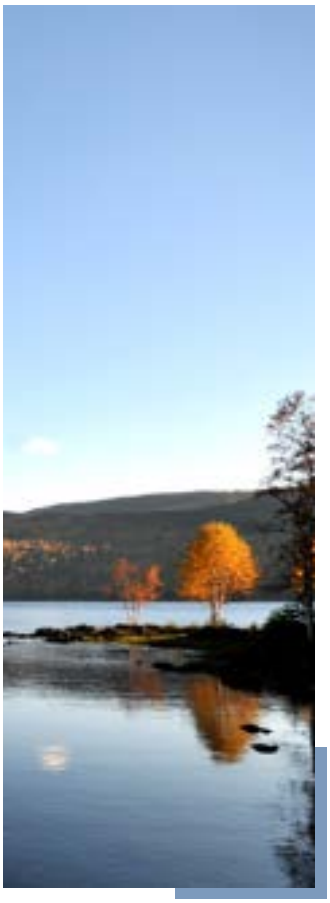
There are also many regulatory requirements which prescribe how customer transactional data can be handled and how private identifying information (PII) should be treated. I prefer to **keep all highly sensitive PII outside of the ledger using tokenization.** For example, tokenize card personal access numbers (PANs) at the edge of the ledger and use the token in the place of the PAN. Unfortunately, as vigilant as your platform may be in the handling of PII you cannot control the upstream and downstream systems and the financial services industry is still rife with systems that are not adequately handling PII. To protect against this unknown, consider data landing zones which are by default encrypted and protected and design PII awareness into the loading of this data from source systems to your ledger.

High Performance Reference Architecture

What is a High Performance Reference Architecture ?

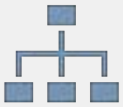
- A modern highly scalable ledger can be constructed using best of breed components
- Sort of a hybrid between build vs buy; build but build with frameworks
- ***TigerBeetle*** for the high performance transactional ledgering
- ***Formance*** as the rich subledger for metadata management, accounting treatment and as the final resting place for financial accounting information
- ***Temporal*** as the integration mechanism ensuring that there is a durable bond between these systems
- This architecture can support 1 million transactions per second with full metadata

As part of this point of view document, I wanted to also share a real world example of how I would design a ledger today which is highly scalable as well as offers the flexibility to handle different types of instruments and treatments - a highly capable ledger is more attainable now than it has ever been given the focus on this area and the evolution of distributed systems and technologies which support transaction durability. My goal is to create a more detailed and specific proof of concept as an example of what's possible. This is not meant as the best or only way to design a ledger architecture but I think it helps to provide a real world example of how the different ledger requirements can be met with existing projects. My general mindset is that I only want to invent what needs inventing so if I can reuse existing projects, frameworks and components I am happy to do so.



For this reference architecture I am looking at three key layers: a high performance transaction layer, a durability and workflow layer, and a layer for accounting treatment and transaction metadata. But why can't this be just one layer or component you might ask? The reason is that specialization and focus makes all the difference when it comes to scalability. For lower throughput workloads which are more latency tolerant you can likely do something simpler.

High Performance Reference Architecture (cont.)



High Performance Transaction Layer

This layer is all about speed - it should only really care about accounts and amounts and being able to get a balance back to a requester as quickly as possible even while adjusting and debiting involved accounts. This is where the core locking logic should exist. For the reference architecture I have included **TigerBeetle** which positions itself as a financial transactions database - only really focusing on accounts and amounts and very little else, but **supporting 1,000,000 transactions per second (tps) which is a formidable benchmark**. *TigerBeetle* has a business model which is open source friendly and this makes it easier to understand how it achieves its impressive performance results and the extent of its capabilities. As a part of my work I developed a utility to help generate accounts, migrate data and interact with *TigerBeetle* called **TigerBeagle**. This open source tool is available to help with testing preparation and migration.

I have tested this platform with 10 million accounts and have found it to be very much focused on exactly the topics of concern for this layer - accounts, transfers, lookups, balance and balance history, and high performance throughput. *TigerBeetle* also offers a few ways to link these accounts to other systems with user data fields and codes and flags which can be used to help identify the account's membership in the chart of accounts.

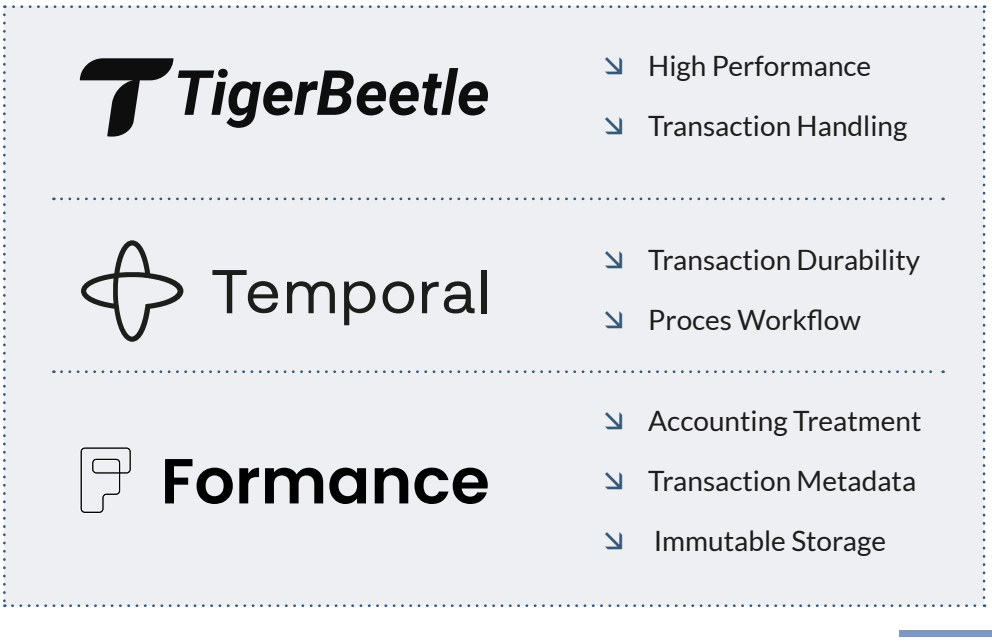
Accounting Layer

This layer is where all of the chaos of payments data is tamed and organized. Each payment type and each payment flow has mappings to the appropriate internal and counterparty accounts. For the reference architecture I selected **Formance**; the rationale is that *Formance* has evolved into more of a financial core platform with fund orchestration, reconciliation, and third party integration built in. Not everything needs to be done in a low latency high throughput way; where *TigerBeetle* is designed to be fast and efficient, *Formance* is designed to be robust and fully featured. Where *TigerBeetle* is good for real time transactions, *Formance* is near-real time and can handle core accounting concepts like end of day and the closing of accounting periods, can emit events to the rest of your systems, offers observability and workflows. *Formance* also offers an API, a CLI and a domain specific language to help with the operational management of financial data – essential for any financial services organization.

High Performance Reference Architecture (cont.)

Durability and Workflow Layer

If you have one layer which can go really fast and another layer which is very robust and can handle all of the accounting logic there is a challenge to keep them aligned – an impedance mismatch of sorts which needs a layer in between to help let the high performance layer sprint ahead while the robust layer catches up in near real time with the required durability and resilience. For this layer I selected Temporal due to its ability to handle complex and challenging process flows with a highly performant and centralized workflow logic.



Conclusion

Throughout this document, we've explored the critical role that well-designed ledgers play in the financial services industry. Drawing from two decades of hands-on experience across hundreds of fintech firms and numerous banks, I have distilled key insights and best practices for creating robust, scalable, and flexible ledger systems.

Key Points Recap

1. **Ledgers as Core Infrastructure**: Ledgers are not just bookkeeping tools; they are the backbone of financial institutions, providing the essential source of truth for all financial transactions. Getting this right is not optional, it's required.

2. **Key Components**: Modern ledgers should consider high-performance transaction handling, proper accounting treatment, flexible metadata management, and immutable record storage as key capabilities.

3. **Architectural Considerations**: The ideal ledger architecture balances performance and flexibility, often utilizing a dual-structure approach with a transactional core and a flexible metadata layer. Supporting transaction durability and recovery is also a consideration - a reference architecture was shared which provides tangible examples on how to design and implement these concepts.

4. **Build vs. Buy**: The decision to build or buy a ledger system should be based on careful cost-benefit analysis, considering factors like unique business requirements and available resources.

5. **Trust and Reconciliation**: Building trust in ledger systems requires comprehensive referential information, double-entry bookkeeping, granular account structuring, and robust reconciliation processes.

6. **Technical Considerations**: High-performance ledgers demand careful attention to account locking mechanisms, transaction sequencing, scalability planning, and failure handling.

7. **Regulatory Compliance**: Ledger design must account for evolving regulatory requirements, emphasizing data clarity, transparency in reporting, and proper handling of sensitive information.

8. **Emerging Trends**: The future of ledger technology is likely to involve blockchain integration, AI-assisted processes, real-time analytics, and increased focus on regulatory technology integration.

Conclusion (cont.)

Future Outlook

As we look to the future, ledger technology will continue to evolve, driven by increasing demands for real-time processing, enhanced transparency, and advances in compute cost performance. The reference architecture presented in this document, combining high-performance transaction processing with robust accounting capabilities and durable workflow management, provides a glimpse into the potential of next-generation ledger systems. To achieve transactions per second scalability similar to this previously required teams of specialist developers and specialized hardware with dedicated data centers and hardware based encryption. Being able to assemble off the shelf components and run this solution on the public cloud is in itself a sign that the future has arrived.

We can expect to see further innovations in areas such as:

- Advanced AI integration for anomaly detection and automated reconciliation
- Enhanced distributed ledger technologies for improved inter-organizational transparency
- More sophisticated real-time reporting and analytics capabilities
- Tighter integration of regulatory compliance features directly into ledger architectures

As financial services continue to evolve, the importance of well-designed, high-performance ledger systems will continue to scale. By understanding and implementing the principles and best practices outlined in this document, financial institutions and fintech companies can build ledger systems that not only meet today's challenges but are also prepared for the complexities of tomorrow's financial landscape. Ledgers keep banks, customers and the industry safe by knowing where the money is and where it has been.

In conclusion, the ledger remains at the heart of financial services. As we've seen, it's not just about tracking debits and credits – it's about creating a robust, flexible, and resilient foundation upon which the entire financial ecosystem can operate and innovate. By continuing to focus on ledger architecture and leveraging emerging technologies thoughtfully, we can ensure that our financial systems remain reliable, efficient, and ready for whatever the future may bring.

Disclaimer

By accepting receipt of this document and reviewing the content set forth herein, you acknowledge having read and agreeing with the following terms.

Certain statements above are based on the subjective views and analysis of Sagard/Portage and cannot be independently verified.

The information contained herein is in summary form for convenience of presentation. It is not complete and it should not be relied upon as such. The information set forth herein was gathered from various sources which Sagard Holdings Manager LP ("Sagard"), the investment manager of Portage Ventures, Sagard Credit Partners, Sagard Senior Lending Partners, Sagard Healthcare Partners and Sagard Private Equity Canada, believes, but has not been able to independently verify and does not guarantee, to be accurate. Sagard makes no representation or warranty, express or implied, as to the accuracy or completeness of the information contained herein. Certain information contained herein has been obtained from published sources and/or prepared by third-parties, including but not limited to companies in which Sagard clients have invested, and Sagard has not independently verified such information. In certain cases such information has not been updated through the date hereof. All information contained herein is subject to revision and the information set forth herein does not purport to be complete.

The attached material is provided to you on the understanding that you will understand and accept its inherent limitations, you will not rely on it in making or recommending any investment decision with respect to any securities that may be issued, and you will use it only for the purpose of discussing with Sagard your preliminary interest in investing in a transaction of the type described herein. Any investment in private markets is subject to various risks; such risks should be carefully considered by prospective investors before they make any investment decision. Each prospective investor should consult its own professional advisors as to legal, tax, accounting, regulatory and related matters before investing.

Sagard is registered as an investment adviser under the U.S. Investment Advisers Act, 1940, as amended. Sagard acts as investment manager of Portage Ventures, Sagard Credit Partners, Sagard Senior Lending Partners, Sagard Healthcare Partners, and Sagard Private Equity Canada.

Sagard Holdings Manager (Canada) Inc. will act as the dealer in respect of purchases of interests by Canadian investors in funds advised by Sagard Holdings Manager LP in the Canadian provinces and territories in which Sagard Holdings Manager (Canada) Inc. is registered as an exempt market dealer. The Ontario Securities Commission is the Principal Regulator of Sagard Holdings Manager (Canada) Inc.

Certain statements and certain of the information contained in these materials represents or is based upon "forward-looking" statements or information based on experience and expectations about these types of investments. The forward-looking statements in these materials include statements with respect to, among other things, projections, forecasts or estimates of cash flows, yields or returns, scenario analyses or proposed or expected portfolio composition and anticipated future events, performance or expectations. For example, such statements are sometimes indicated by words such as "expects", "estimates", "believes", "forecasts", "seeks", "may", "intends", "attempts", "will", "likely", "should" or negatives thereof and similar expressions. Forward-looking statements are inherently uncertain and are not guarantees of future performance and are subject to many risks, uncertainties and assumptions that are difficult to predict. Therefore, actual events or results or the actual performance of Portage Ventures, Sagard Credit Partners, Sagard Senior Lending Partners, Sagard Healthcare Partners, and Sagard Private Equity Canada may differ materially from those reflected or contemplated in such forward-looking statements as a result of various factors. No representation or warranty, express or implied, is made as to any forward-looking statements and information and no undue reliance should be placed on such forward-looking statements and information. Sagard has no obligation and does not undertake to revise or update these materials or any forward looking statements set forth herein, except as required by law. In addition, unless the context otherwise requires, the words "include", "includes", "including" and other words of similar import are meant to be illustrative rather than restrictive.

The information in the attached materials reflects the general intentions of Sagard. There can be no assurance that these intentions will not change or be adjusted to reflect the environment in which Sagard will operate. Certain statements in these materials contain prior performance indications. Past performance and historic information is not necessarily indicative of future activities or returns, and there can be no assurance that Sagard will achieve comparable results. Conclusions and opinions do not guarantee any future event or performance. Neither Sagard nor any of its subsidiaries or affiliates are liable for any errors or omissions in the information or for any loss or damage suffered.

The materials contained herein are for information purposes only and do not constitute an offer to sell or a solicitation of an offer to purchase any interest in any investment vehicles managed by Sagard.

No securities commission or regulatory authority in the United States or in any other country has in any way passed upon the merits of an investment in the Fund or the accuracy or adequacy of the information or material contained herein or otherwise. This information is not, and under no circumstances is to be construed as, a prospectus, a public offering, or an offering memorandum as defined under applicable securities legislation. The information contained herein is intended solely for "qualified purchasers" as that term is defined in the U.S. Investment Company Act of 1940 and, in Canada, "accredited investors" within the meaning of applicable securities legislation. Such offer or solicitation shall be made only pursuant to a confidential private placement memorandum or similar document, which qualifies in its entirety the information set forth herein and contains a description of the risks of investing. The attached material is also qualified by reference to any limited partnership agreement or similar document and subscription agreement relating to a Sagard-managed investment vehicle. All of these other documents relating to a Sagard-managed investment vehicle should be reviewed carefully prior to making an investment.

This document, which has been prepared solely for information purposes by Sagard, is confidential and is being provided to you on the express understanding that it will not be reproduced or transmitted by you to third parties without Sagard's prior written consent. Without limiting the foregoing, you (and your employees and agents) agree that you will keep the information contained herein as provided herewith confidential and agree that you will, and you will cause your directors, partners, officers, employees, professional advisors and representatives, to use such information only for information purposes and for no other purpose and will not divulge any such information to any other party. If you are not the intended recipient of this document, you are hereby notified that the use, circulation, quoting or reproducing of this document is strictly prohibited and may be unlawful.

Additional information is available upon request.

All information is presented as of November 23, 2024 unless otherwise stated.

Sagard

Montreal

1172 Sherbrooke Ouest

Montréal, QC H3A 1H6

Canada

+1 (514) 286-6248

Toronto

161 Bay Street, Suite 5000

Toronto, ON M5J 2S1

Canada

+1 (416) 607-2250

New York

280 Park Avenue, 29th Floor East

New York, NY 10017

United States

+1 (212) 380-5605

Paris

49/51, Avenue George V

75008 Paris

France

+33 (0)1 53 83 30 00

Denver

1099 18th Street, Suite 2900

Denver, Colorado 80202

United States

+1 (303) 986-2222

Naples

4850 Tamiami Trail North, Suite 301

Naples, FL 34103

United States

Abu Dhabi

Al Sila Tower, ADGM Square, Unit No. 1, Floor 12

Al Maryah Island

United Arab Emirates

+971 2 411 6239