EVENT STORE®

How a new category of database is **transforming the fintech industry**

Contents

Key takeaways

- Every five years, a new category of database appears—consider the impact hierarchical, relational, graph, and document data models have had on the operational database market—and the financial services industry is ripe for its next disruption.

- Microservices, data analytics, artificial intelligence (AI), data mesh, machine learning (ML), and blockchain are market forces pushing financial companies to become data-led and transform their core databases.

- A category of database has evolved from the Event Sourcing community. Unlike traditional databases, this databasel doesn't overwrite your data, so no data is ever lost. Each state-transition is stored in a sequence that can be read or questioned as needed, with streaming access provided. You can move up and down the stream of events, giving you new insights and the power to make better, more informed decisions.

- Financial companies are heavily regulated and looking for faster and more proactive database solutions. This new category of database is the gold standard for audit and compliance, allowing for synchronization across microservices and downstream data sources while responding to new business requirements like no other database technology.

Introduction

Are you a software architect in the financial services industry motivated to learn about a new category of operational database? Then you've come to the right place.

We know the financial services industry is undergoing enormous technological and regulatory change, with heightened pressure from watchdogs and consumers alike.

> "
>
> Three out of four banks and credit unions have embarked on a digital transformation initiative

More than ever before, data is a critical asset, especially as the finance industry has heavy regulatory and audit requirements and stores enormous datasets. According to Deloitte's 2022 banking regulatory outlook report, data plays a leading role in identifying emerging risks and developing risk mitigation resiliency—vital during economic uncertainty.

For decades, databases have been essential for storing data in a structured form and tracking banking applications' transactions, balances, deposits, loans, credit cards, engagement with banking apps, calls to service centers, and visits to branches.

> "
>
> If data is the fuel of the digital economy, the operational database is the engine
>
> **— William Blair and Company, 2021**

The truth is, new categories of database technology don't come around that often—but when they do, they have a profound effect. The relational, graph, and document data models all had a mammoth impact on the finance industry: Today, practically every financial business is using traditional Relational database technology like Oracle or Microsoft SQL Server, Document databases like MongoDB, and others.

But more and more software architects are finding that traditional database technologies can no longer support the real-world and real-time demands of modern-day financial systems.

Leading financial companies are facing downstream complexity, losing critical data, and not storing the business context. In an attempt to make up for critical historical data not being stored, they have been pushing data into complex extract, transform, and load (ETL) pipelines, data lakes, and warehouses, where the data loses value and changes meaning over time.

It's time for the next disruptive database.


```
// Run commands in an elevated administrator shell

$ choco install eventstore-oss
Are you sure you want to install 'eventstore-oss'?
(y/n) y
■■■■■■■■■■■■■■■■■■■■■■ 100%
Event Store successfully installed!

$ EventStore.ClusterNode.exe
■■■■■■■■■■■■■■■■■■■■■■ 100%
Event Store started!
Access the Admin UI on http://localhost:2113 (admin:changeit)

// You're ready to start writing events! Click the 'Get
started' link for guidance on creating a demo application
```

A new category of database technology **for a richer data story**

At Event Store, we recently analyzed our top fintech clients overwhelmed by the minefield of database options. We know many more software architects at financial companies are out there creating new applications and considering new data models—all expecting more from what's currently available.

At the most fundamental level, storing data—without losing any of it—is absolutely essential. But financial companies also need to organize and access their data efficiently, so they can quickly understand customers, predict trends in customer behavior, and tailor interactions.

That's why many fintechs and financial companies have opted for a new database technology, which maintains the fast distribution of information, keeps an audit trail of who did what, and helps meet regulations.

Instead of saving just the current version of the entity state into a database, this new data model stores each state transition and names it as a separate record called an event.

> "This means the order and business context of state changes are kept, transforming modern enterprises' data into real-time actionable streams. We implement this "source of truth" database at Event Store, and it is a confluence of two macro trends: the evolution of operational database technology and event-driven architecture (EDA)"
>
> **— Dave Remy, CEO of Event Store.**

It keeps data at its most fundamental level, streams it in real-time, and transforms data into any other shape needed by downstream data technologies. When business requirements change or new analytics technology emerge, it allows developers to replay from the beginning of time. EventStoreDB forces developers to analyze and address the impact of data changes on historical data, not just to the current state data like traditional databases. This is fundamental for enterprises to take advantage of the explosion of innovation in analytics and AI.

The new database **(EventStoreDB)** consists of:

The State-transition data model

Transformation

Streaming (real-time/replay)

On the shoulders of giants: How this new category of database emerged from **Event Sourcing**

The Event Sourcing community has been active since before 2012 when EventStoreDB first went into production. Event Sourcing is a design pattern and its main focus was building software applications. There was little focus on a profound aspect of Event Sourcing that proposed a new, powerful data model underlying modern software applications.

Event Sourcing practitioners originally saw Event Store as a place to persist the state of their application. This state was stored as a sequence of named state transitions, a revolutionary new way of storing data. However, the implications from a data perspective, in addition to the application development benefits, were rarely emphasized.

Event Store was one of the first implementations of this new data model. It represents the next generation of the operational database, an area where there has been very little innovation in the last 40 years. It represents a shift in thinking about what an operational database should do - first and foremost fully capture data at origin, the moment of truth at the very beginning of the data lifecycle. Current operational databases are not fit for purpose.

EventStoreDB is uniquely relevant to practically every key software trend affecting financial institutions today. From microservices, analytics, artificial intelligence, distributed systems, and blockchain. Because it keeps data at its most fundamental level and streams it in real-time (or replaying from any point in time). Enterprises can respond with speed and agility when business requirements change. Because each state transition allows businesses to trace how events occur and why, the gold standard for audit and compliance.

So, if you are responsible for the technology and integration roadmap for a financial company or startup, this is a rundown of how this new data model is disrupting the fintech industry and how to jump on the bandwagon.

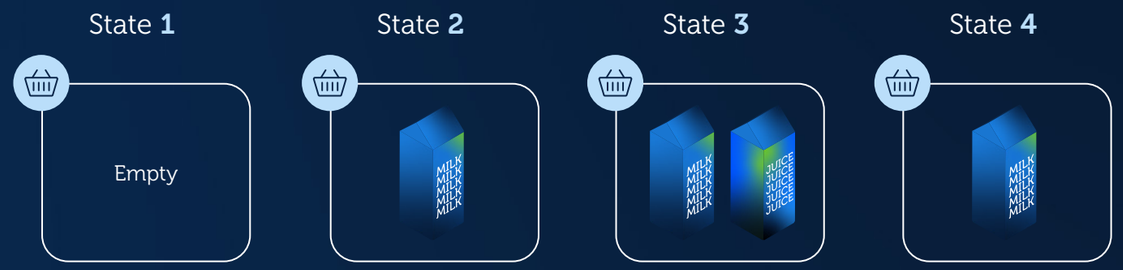
Storing the *why* to turn data **into limitless business insights**

So, you may be wondering: “But if I store all the events, doesn’t that lead to clutter? Why would I want to do that? And why not stream events instead?” Well, when developers come across a data problem, they want to work out how they got to that state. But it’s hard to track backward with old data models where events have not been stored. They need an operational database with real-time events storage.

Financial companies are now after an easier way to see the why of an immutable sequence of changes—and a way for no data to be lost, overwritten, or unavailable. This new data model is a simple yet powerful concept that records not just what’s changed, but the order those changes occurred. This creates a permanent, unchangeable log of all the transitions the data has gone through.

Take this simplified example of a shopping basket:

1. The state of this shopping basket is empty.
2. I add a milk carton to my basket, so the state changes.
3. I add juice too, and the state changes again.
4. Then, I decide I don’t want to buy the milk and take it out. I only buy the juice.



Storing the *why* to turn data **into limitless business insights**

What's more valuable? Knowing only what the customer bought (the end state), or every single event that led up to it?

"

Most developers are trained to look deeply at the system they are building and care less about what's happening on the outside. But we are storing the why, so the database becomes business-focused. Developers need to be interested in their company's trajectory.

— Yves, Head of Developer at Advocacy

By storing data as a state transition event with a sequence number, worlds of possibilities appear: A developer with any technology stack can find where the events are in the stream of changes. By knowing that, events can be replayed from any point in the sequence, which could help with calculating pricing models and updating inventory lists in a financial company.

The idea is far from new! We can find examples from many environments—like a bank account ledger. Our current account balance could be represented as a number and a currency (10 EUR) or as the result of the sum of all the transactions (0 EUR + 20 EUR + (-15 EUR) + 5 EUR).

Also, with this new category of database, streams can be transformed into other streams that serve as input for app-specific integration, enabling businesses to maximize the value of event data and use it to boost business processes.

You can also inject new analytic methods into old data to find new insights. Imagine using your analytics engine on years of old data and discovering your business could have been more effective and profitable with some minor changes.

If you are unsure what the advantages of EventStoreDB are over traditional databases, or you've had trouble convincing your CTO to update legacy systems, we're here to help.

EventStoreDB: How it differs from existing operational databases used in financial services

Microservices, data analytics, AI, ML, data mesh, and blockchain are market forces that demonstrate the transformation to a data-led business community. Therefore, the requirements of, and even how, architects and developers think about operational databases are changing. With the operational database market set to grow to $60.8 billion in 2025, what can we learn from existing operational databases to enhance future data models?

A walk through time: The database evolution

Greek Philosopher Heraclitus said, "the only constant in life is change." And if we're expecting constant updates, then reviewing how we got to where we are today will help us to learn and grow. So, let's have a quick recap:

1. **Hierarchical databases** — IBM (1968)

In 1966, IBM faced an important question: If we could put a man on the Moon, could we also create a computer program to track the millions of rocket parts it takes? Their response arrived two years later in the form of the world's first commercial database management system, named Information Management System (IMS), in 1969.

Many of the largest corporations in the world still rely on the system to run their everyday business. The hierarchical, tree-like structure data model and transaction-processing software handle complex, high-volume transactions, such as financial applications, inventory management, and airline reservations.

2. **RDBMS** — Oracle & Microsoft (1977)

Another trendsetter is Oracle Database, a relational data model (RDBMS). The landmark event in history was in 1979 when users could describe in Structured Query Language (SQL) what they wanted to do, and the SQL language compiler could automatically generate a procedure to perform the task—known as the SQL-based RDBMS.

RDBMS store data logically in schemas and physically in data files (tables and indexes). This separation allows the management of physical data storage without affecting access to logical storage structures.

3. **Graph data model** — Neo4j (the mid-2000s)

Next up: Graph databases. With Neo4j and Oracle Spatial and Graph, you don't have to declare your schema intentions ahead of loading data. This allows you to get your data in as quickly as possible and then test hypotheses right away.

A graph database consists of entities called nodes, and the connections that join them together are called relationships. The query language Cypher focuses on pattern matching. Unlike RDBMS, there's no more worrying about JOIN tables and foreign keys.

4. **Document data model** — MongoDB (2009)

In 2009, the market was ripe, and MongoDB launched its document (or NoSQL) database, significantly disrupting the data world. Instead of storing data in fixed rows and columns, this database uses documents.

Documents store data in field-value pairs. The values can be a variety of types and structures, including strings, numbers, dates, arrays, or objects. Documents can be stored in formats like JSON, BSON, and XML.

Interjection point: Event Streaming **vs.** EventStoreDB explained

Another disruptor to the finance industry is Confluent, a platform with a cloud-native design that helps developers build and scale real-time applications. Founded by the original creators of Apache Kafka, an open-source distributed streaming system, Kafka is not a database technology. It is about data in motion, not the long-term storage of data.

Questioning the difference between Confluent Kafka and EventStoreDB? When you think of Kafka or Event Streaming, imagine a live webinar: You view it (your data), and then it's gone. EventStoreDB is on-demand and stores the stream of events so that you can pause, fast forward, and replay data for future analytics.

Event Streaming is crucial in financial-trading applications where actions have a time-bound window and require immediate action. But, when it comes to financial industries with heavy regulatory and audit requirements, a State-Transition data model is a must to trace those streams.

Event Store 2023

Data plus architecture is key to the success of the modern enterprise. In the last few years, data architecture lost its glamor for many software architects. Document data models were a big part of this: *Throw your data into a "document" and worry about it later* became a popular approach. This is kind of insane. Data is the raw material for an information enterprise. Not being a proper steward of that data is negligent.

Dave Remy, our CEO, who once adopted EventStoreDB for a major transformation project for one the largest global asset management software companies.

RDBMS vs EventStoreDB: What's the difference?

	EventStoreDB	Relational DB (RDBMS)
Streaming data	When an event happens, it is stored and then broadcasted directly. Events can be replayed from any point into streams. And streams can be combined into other streams and persisted in the database. Downstream data systems can know where they are in the stream of changes and stay synchronized.	Traditional databases don't stream by default and require add-ons to support streaming. Even then, they only stream out-of-context changes. They were built to hold data until it is queried rather than streaming it in real-time to all the downstream users.
Efficiency	Saves changes once, subscriptions communicate changes instantly and asynchronously throughout the enterprise.	"Traditional databases pool data; when you want the data, you go to the database and pull it out. Batch processes to extract, transform, and load the data into other data sources overnight are still common approaches for dealing with these databases."
Service autonomy	Caches are one of the best practices for service autonomy. With EventStoreDB, caches are synchronized using the sequence number, allowing services to be independent if the source service goes down.	Synchronization of caches is very difficult.
Event-driven architecture (EDA)	Builds complex workflows and reacts in real-time. When an event is sent out over Kafka, for example, or if there is an issue or business insight, EventStoreDB can trace the originating event in the system (as opposed to a database that "forgets").	Relational databases have no first-class concept of events. There is an inherent mismatch with EDA as they forget the past.
Time travel	Developers can move through the event streams in any direction for analysis and forecasting. Sequencing is an integral part of this data model.	Not built-in as part of the data model, changelogs might be needed for specific use cases. While other databases can have a notion of time, temporal table needs to be explicitly enabled.
Root cause analysis	Traces events back to their roots easily with sequence numbers, making analysis easy and faultless.	Developers have to refer back to changelogs, check external data sources, and piece together information.
Legacy migration	EventStoreDB is built for connectivity (and more granular). Therefore, pieces of legacy systems can be pulled out and incrementally rewritten. Also, this database can feed back the data a legacy system needs in real time.	"Needs long-term planning, downtime, and to happen all at once. Change is hard as the impact on other (sub) systems is largely unknown."

The solutions to the finance industry's pain points

Storing every relevant event

There's value in documenting successes and errors, so there isn't a clear distinction between relevant and irrelevant data.

By viewing all events in a timeline, developers will know why decisions were made and learn from them. Also, seeing the progression of events and visualizing how an error developed aids in tackling its root cause and prevents it from repeating itself.

Let's go back to an old, classic example: An air conditioning unit. It constantly measures the temperature. If you were using EventStoreDB, and the temperature starts suddenly climbing, the system will store this shift and why. The unit would notice a change in external temperature or that someone had manually changed it.

Flexible automation of decisions

Traditional relational systems automate complex work from the get-go. But, put into practice, it doesn't always make financial sense to automate processes without knowing that the business truly needs it. Companies must identify if automation is cost-effective before setting the process in motion.

A State-Transition data model defers decisions to automate tasks. Companies can perfect their operations, gather data from their performance, and then **automate from the moment they can prove it's cost effective.**

This new database grants you the flexibility to make changes whenever you need, not just strictly from the beginning.

Straightforward auditing

Financial institutions must follow strict regulations because of their data's sensitive nature. IT teams understand these requirements and must figure out how changelogs operate in their databases so there's no missing information. The State-Transition model can spare teams of these worries since it is not possible to change the system without that change being represented in the database.

Auditing is about transparency at the highest level. But fintechs often count on unreliable record-keeping with traditional databases that don't store all data to perform audits confidently.

With the State-Transition model, all your data is stored from A to Z, leading to perfect and effortless audit trails.

Event Store 2023

An urgency to meet fast-changing and evolving consumer demands has led financial companies to become more innovative and fintechs to scale. The State-transition model is here to bring the sector's databases up to speed.

Future-proof databases with 'what if' scenarios

Integrating a State-Transition-database means you don't need to wait to track and trace any internal issues; you can plan for them.

Within a controlled environment, you can rewrite the language code to mimic a bug or new functionalities to see how your software responds. **This use of trial and error and 'what if' scenarios will save businesses major headaches by predicting outcomes.**

Users can make changes and they can be replayed on historical data, so differences can be seen and understood

Successful language integrations

EventStoreDB is widely supported by many programming languages. It uses gRPC to communicate between the cluster nodes and client servers. When developing software that uses EventStoreDB, we recommend using one of the official SDKs:

- .NET
- Java
- Node.js
- Go
- Rust

Better security

Financial companies handle personal data, so their databases must comply with international accreditation standards.

The 2022 Event Store Cloud is now an ISO/IEC 27001:2013 certified service, as verified by independent auditors A-LIGN.

Event Store Cloud is also SOC 2 Type 1 compliant, which evaluates a company's cybersecurity controls, determining whether customers' data is sufficiently protected.

These certifications will help you when regulators, business partners, and suppliers are looking to work with businesses that have credible database security frameworks.

Know you are following GDPR

EventStoreDB still comes under the same guidelines as any other storage technology within the industry and is subject to the same rules regarding managing data.

If a consumer was to submit an Article 17 request for data erasure, then the database can separate private information from the rest. You can forget one stream of information without impacting the whole system.

Suppose a user named Jennifer wished to remove her personal data. In that case, all her personal information could be deleted from her data trail. Her profile would remain but could be named something ambiguous, like user 1, and the system would still have a log of all past and current events.

How to get started and successfully write an application with **EventStoreDB**

EventStoreDB is open source as we'd prefer our product to belong to the developer community, who can add to it and keep evolving with the product and piloting projects. To be the operational database of the future, the infrastructure needs to be accessible worldwide.

Get Started Guide

Event Store 2023

However, if you are in need of end-to-end database support, we have the Enterprise Support Subscription and Event Store Cloud, managed by the Event Store team. It works across all three major cloud vendors, allowing customers to choose their region with their chosen cloud provider, which is essential to keep latency acceptable in an operational database scenario. The basic functionality of selecting a region, a cluster size, provisioning the cluster, and automatically deploying it are all included in the product and used on a daily basis by our customers.

Enterprise support subscription

End to End Database support.
Tiered pricing by SLA plus per cluster charge.

Event Store Cloud

Managed EventStoreDB by the team that built the database.
GA on June 8, 2021.
All major cloud providers.

Enterprise Edition (coming soon)

License/subscription model for enterprise features (e.g., GeoReplication, Advanced Security, Management Console, Private Ledger, etc).

Training

Public and private classes on Event Sourcing and EventStoreDB.

Consulting

Advisory consulting.
Consulting partnerships.
Transformation services.
Implementation services.

EVENT STORE