Trading Systems Powered by XTDB

Every trading platform must now deal with immutable, bitemporal data. However, the demands of the middle office far exceed the capabilities of early 21st century data storage techniques.

Trade capture, trade and positions queries, P&L, risk calculation, and instrument reference data must all understand the difference between "knowledge time" and "trade time" or quants cannot analyze the truth of market data from disparate sources. XTDB has native bitemporal records so your true trade data is never lost.

XTDB Sector Brief - Trading Systems

1. Executive Summary

Across the billions of significant daily events, across a firm's diverse teams, many trading systems engineers find themselves forced to deal with the convergence of three data paradigms: classic relational data, event streams, and downstream analytics. Historically, tying these three types of data together required cobbling together event sources like Kafka or RabbitMQ with traditional RDBMSes like Oracle or Microsoft SQL Server. Quantitative Analysis was (and is) often executed out-of-band, requiring system owners to funnel live data into time series databases, generating stale copies. Along the way, valuable information, such as the "knowledge time" of a particular trade, is easily lost between these systems.

Every record in XTDB is immutable and contains two temporal coordinates, ensuring *time* is always recorded. Although XTDB treats events as first-class, the XTDB query engine does not suffer from the problems of querying event streams such as rebuilding aggregate states from the epoch or caching snapshots. The XTDB query engine understands that users always want answers "*as-of now*" by default and that historical immutable records are not to be considered unless explicitly requested by software intentionally manipulating the temporal plane. Because *time* is a first-class citizen in XTDB, Quantitative Analysis nodes can operate on a verbatim copy of realtime data.

2. Background

It was a struggle to explain the concept of a corporation to the average British citizen in 1601. Whether joint stock companies or algorithmic trading, a new idea leaves humanity gravid until its social acceptance — and the birth is often forced. Over a period of three hundred years, humanity experienced the genesis of companies, stocks, exchanges, and the telephone. In a period measured in decades, trading firms have learned to contend with computers, the internet, nanosecond execution windows, and the limits Physics places on human creation.

Trading firms have always adapted to changes in tooling and environment, either by force or by well-timed predictions. From James Sterngold's 1984 article *The Anatomy of a Stock Trade* [1]

> Michael Creem, an exchange vice chairman and floor specialist, said, "You will see dazzling improvements in the way information is delivered to brokers and reporting, but decision-making by people is essential to the system. That won't change."
>
> — Anatomy of a Stock Trade

The idea that any task of the human intellect is somehow precious and incapable of automation seems archaic to us now, but the milestones of automation remain difficult to predict. More often than not, automation is adopted by force. Many human traders still have jobs and the older ones are even likely to retire in their field.

Later in Sterngold's article, however, he foreshadows a twenty-first century economy obsessed with *time*:

> Barbara Yohle paused to pencil in a few bubbles on a computer card and dropped it in a slot behind Mr. LaBranche reporting the trade. The process, which began in Helena, Ark., and ended up with a confirmation at the exchange at 11 Wall Street took just over five minutes.
>
> — Anatomy of a Stock Trade

In 1984 — a year when stock trades were still conducted with pencils and telephone calls — a five minute interval was as good as instant. The article goes on to describe the clearing process, which took days. "*When did the trade occur?*" is an unanswerable question. Even in 1984, a trade lived on at least two timelines: the human's and the machine's.

In 2022 — a year when entire markets are won and lost on nanosecond performance — a five minute interval might as well be an eon. In forty years, the most important change to the execution of a trade is the scale on which it occurs.

3. Problem

Each team wants something unique from the firm's trading infrastructure. Algo and grey-box desks maintain local copies of trades and positions relevant to them, which means they often make a large beginning-of-day (BOD) query to central trade infrastructure. At a similar scale, in a completely different discipline, desks or firms engaged in slower interday trading or investment often batch risk calculations during their end-of-day (EOD) processes. Near-realtime trade and positions streams are needed by intraday front office systems and realtime risk calculation. Even a simple constraint, such as a trading desk demanding read-your-writes consistency from the trade store, demands low-latency streaming. Quants are known to repeatedly request "all the trades, ever" — a simple query deserving of a simpler interface than an API call.

Each of these teams is interlinked and it matters a great deal that the data they build their architecture and algorithms against is consistent. However, *consistency* is a moving target. The temporal complexity of 1984 is exacerbated by an increasingly sophisticated environment. Each entity in a network of trading systems now enjoys the temporal nuances only the *trade* enjoyed in 1984. Every interaction in the trading system is now an event: capture, query, position calculation, risk calculation, clearing, and instrument lifecycles.

As Thomas Pierrain discovered while working on a middle office instrument reference data system, product owners are now asking questions that many systems cannot yet answer. [2]

> Can we produce a report in the past — but at a given value date? Can we fix the past? Can we schedule [instrument creation] events in the future? What about performance? Could we search all instruments matching a bunch of properties but at a given value date … in less than a second?
>
> — Thomas Pierrain

The discussed solution leans on event streams, which affords the firm the ability to record and observe time for both the human and the machine (known as bitemporal data). [3, 4] However, event streams present their own difficulties. How does a system query an event stream without replaying all events from the beginning of time? How can a system replay trades into an event stream for the purposes of back-testing, calibrating algorithms, analyzing historical patterns, or simulating future patterns? How can a firm's quantitative analysts query across time itself to answer bitemporal questions, such as "how would my algorithm have performed as it existed on April 23rd if it had market data as of May 7th?"

Event streams are an important stepping stone toward trading systems with fully bitemporal records across all interactions, across all teams, but they are not enough on their own. Such systems require a powerful bitemporal query engine.

The world's banking systems have understood the value of immutable ledgers for a very long time. [5] Yet, it is not uncommon to find trading firms, investment banks, and hedge funds shoehorning an RDBMS with a fundamentally mutable character into these roles with the use of append-only tables and event source plumbing. These twentieth century databases cannot provide core bitemporal data, however, and artificial layers intended to compensate for this shortcoming only multiply the complexity of trading systems.

4. Solution

Every record in XTDB is immutable and bitemporal. Because XTDB is built from the ground up with both these concepts in mind, querying XTDB is as natural as with any traditional database; it assumes a default value of "now" for all timestamps, whether a trade or instrument is being recorded or queried. Only in the increasingly-common cases where a trading system is explicitly interested in the history of a trade does the time-traveling surface area of XTDB show itself.

XTDB automatically tracks both *Application Time* (otherwise "*Trade Time*" or "*Valid Time*") and *System Time* (otherwise "*Knowledge Time*" or "*Transaction Time*") for every single record. Time-traveling queries in XTDB are as natural as any other query: "as-of now" queries require no special parameters and "as-of *{timestamp}*" queries require no more than one parameter for Application Time or System Time. Time-traveling, bitemporal "as-of" queries are available by specifying both.

Legacy databases struggle with the downstream consequences of immutable, bitemporal data and time-traveling queries. Temporal data implies that any entity can be viewed at any point in its history. This is exceptionally difficult to model with a *schema-on-write* database such as Postgres or Microsoft SQL Server because the schema must be permitted to evolve naturally, as the entity history evolves. Some digital trading systems have records reaching back to the early 1990s — had those trades been recorded *as they were* in 1992, they would not have the same schema as trades recorded thirty years later. Rather than manipulating historical data into a shape it never had when it was recorded, XTDB provides *schema-on-demand*. Just as a JSON document does not carry its schema with it, trading application tiers are free to apply schema to entities returned from an XTDB query.

Whether a trading firm employs a service-oriented architecture or a monolithic architecture, relationships between trade data exist. Services define relationships by API. Monoliths define relationships intrinsically. Both architectures are comfortably supported by XTDB. Since immutable records with a full bitemporal history cannot exist in a network of static schema, relationships in XTDB are implicit: any two entities with attributes of equal values are related to one another and can be joined. [6]

There is risk inherent in every software system a trading firm decides to own. Each system is a new network of dependencies. Immutable trade history has been the status quo since the early computerization of markets. However, most firms today are building bitemporal systems from scratch to support their trade infrastructure. XTDB brings immutability to the firm's entire data set and eliminates the risk of owning ad-hoc bitemporal systems.

```
SELECT options.strike_price FROM options
  WHERE options.ticker_symbol = 'TSLA'
    AND options.year = 2020
    AND options.month = 12
    AND options.call_put = 'C';
```

This example SQL query will look very familiar to users of relational databases. Every entity carries its entire history, which can be queried on both timelines, but only the latest immutable record is returned from the `SELECT` statement. Neither timeline is constrained, so both are assumed to be *now*.

5. Future Work

As of this writing, XTDB 1.20.0 is a full-featured bitemporal database providing as-of queries in a wide range of domains. (See: https://xtdb.com/solutions/) Ongoing research and development is expanding the core pillars of XTDB. This new research enables a full bitemporal index capable of answering queries *across time* in addition to time-traveling queries with as-of semantics. [7, 8] Due to the disk volumes required to store and process fully immutable records, the new architecture also enables the separation of storage and compute (SoSC). [9, 10, 11] Underpinning SoSC is a new data model built on Apache Arrow, which enables seamless interop between trading desks, middle office, quants, and data scientists. (See: https://arrow.apache.org/) Finally, the XTDB SQL engine is undergoing a ground-up rewrite to support native ANSI SQL:2011. The new SQL engine is designed to precisely interact with XTDB's immutable, bitemporal, schema-on-demand records and indexes.

6. References

[1] J. Sterngold, "Anatomy of a Stock Trade." The New York Times, 1984, [Online]. Available: https://www.nytimes.com/1984/08/09/business/anatomy-of-a-stock-trade.html.

[2] T. Pierrain, "As Time Goes By... (a Bi-temporal Event Sourcing story)." 2018, [Online]. Available: https://2018.dddeurope.com/speakers/thomas-pierrain/#talk2.

[3] M. Kleppmann, *Designing Data-Intensive Applications*. Beijing: O'Reilly, 2017.

[4] M. Fowler, "Bitemporal History," 2021, [Online]. Available: https://martinfowler.com/articles/bitemporal-history.html.

[5] M. Smith, "Luca Pacioli: The Father of Accounting," *Available at SSRN 2320658*, 2018.

[6] C. Rost, P. Fritzsche, L. Schons, M. Zimmer, D. Gawlick, and E. Rahm, "Bitemporal Property Graphs to Organize Evolving Systems." 2021.

[7] R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[8] C. S. Jensen and R. T. Snodgrass, "Temporal Data Management," *IEEE Transactions on knowledge and data engineering*, vol. 11, no. 1, pp. 36–44, 1999, doi: 10.1109/69.755613.

[9] M. Vuppalapati, J. Miron, R. Agarwal, D. Truong, A. Motivala, and T. Cruanes, "Building An Elastic Query Engine on Disaggregated Storage ," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, Santa Clara, CA, Feb. 2020, pp. 449–462, [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/vuppalapati.

[10] B. Vandiver *et al.*, "Eon Mode: Bringing the Vertica Columnar Database to the Cloud," in *Proceedings of the 2018 International Conference on Management of Data*, New York, NY, USA, 2018, pp. 797–809, doi: 10.1145/3183713.3196938.

[11] L. Bindschaedler, A. Goel, and W. Zwaenepoel, "Hailstorm: Disaggregated Compute and Storage for Distributed LSM-Based Databases," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2020, pp. 301–316, doi: 10.1145/3373376.3378504.