

[Home](#) > [Blog](#) > [Data Engineering](#)

Time Series Database (TSDB): A Guide With Examples

Learn about time series databases and how they differ from traditional databases, as well as their key features, use cases, and popular options.

[☰ Contents](#)

Updated Feb 21, 2025 · 8 min read



Marie Fayard

Tech Lead & Product Strategist

TOPICS

[Data Engineering](#)

A couple of years ago, during my first week in a new software engineering role, I was asked to investigate **time series databases (TSDBs)** to replace our Postgres solution.

I knew absolutely nothing about the topic and had so many questions. What even is a time series database? How does it work? How is it different from a [traditional database](#)? Why should we use one? Do I need specific skills for that?

Since then, I've learned a lot about TSDBs and have applied this knowledge in various companies to solve a wide range of problems.

In this article, I'll sum up what I learned over the past few years to give you a good idea of what TSDBs are, how they work, and what use cases they best work for. I'll also walk you through some of the TSDBs currently on the market and give you tips so you can choose the one that best suits your needs.

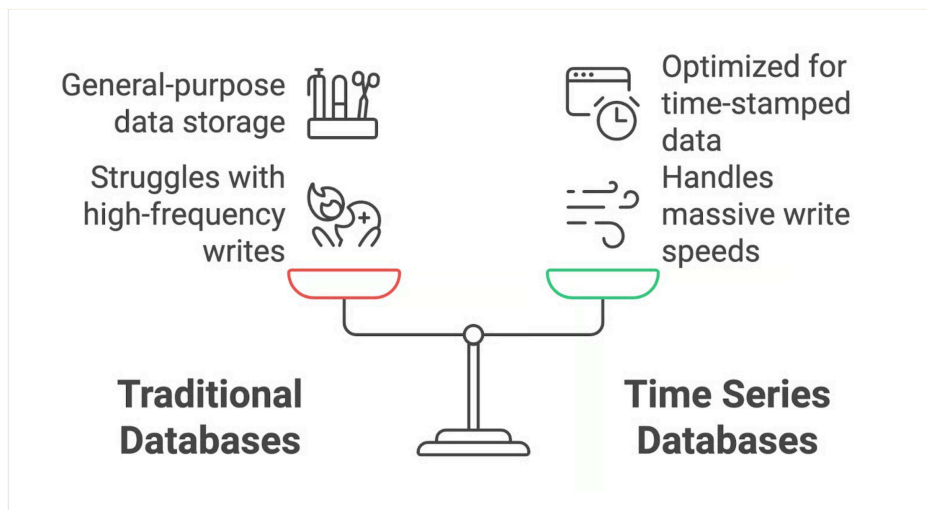
What Are Time Series Databases?

Imagine a smart thermostat sold by company X that records temperature readings every 30 seconds. In a single day, this one device generates thousands of data points. Now, multiply that by hundreds or thousands of devices across a city, and the volume of time-stamped data gathered by company X becomes staggering.

To store this data efficiently and analyze trends (like temperature changes over time or sudden spikes), company X needs a database that can handle massive write speeds and perform time-based queries efficiently.

Traditional databases tend to struggle with this kind of workload because they aren't designed to handle high-frequency data writes or query data efficiently over specific time ranges. That's where time series databases come in.

Time series databases are specialized databases designed to manage data that is organized and indexed by time. Unlike traditional databases, which are optimized for general-purpose data storage, TSDBs focus on efficiently storing, querying, and analyzing sequences of time-stamped data points.



TSDBs are particularly suited for applications that deal with continuous **data streams**, such as IoT, **DevOps** monitoring, and financial analytics.

Characteristics of Time Series Databases

There are a few things that TSDBs do differently than traditional databases.

Optimized for time-stamped data

At their core, TSDBs are built to handle data with timestamps as a fundamental attribute. Every data point in a TSDB includes a timestamp, which serves as its primary index. This allows these databases to efficiently store and retrieve time-ordered sequences and provide quick access to historical trends or recent events.

Most TSDBs use *time-based partitioning*, meaning the data is stored in partitions based on time intervals (e.g., hourly, daily). This enables efficient pruning, where queries ignore irrelevant partitions altogether.

They can also implement *time buckets*, grouping data into predefined time windows (e.g., 1 minute, 1 hour) for faster aggregations.

High ingestion rates

Time series data is often generated at a rapid pace—think of IoT devices sending thousands of data points per second or a server monitoring tool capturing system metrics in real time. TSDBs are optimized for these high write rates and can ingest vast amounts of data without slowing down or losing information.

This is usually achieved using append-only data storage models and in-memory buffers to prevent locks or transactional bottlenecks.

Efficient queries for time ranges

Analyzing time series data often involves querying specific time intervals or windows, such as “last 24 hours” or “this year compared to last year.” TSDBs are built with this in mind, offering specialized query capabilities that allow users to quickly retrieve data over defined time ranges. They also support aggregations like averages, sums, or trends to offer valuable analytics without complex query logic.

The query optimization techniques include:

- **Pre-aggregated data:** TSDBs often pre-calculate summaries for common time intervals (e.g., hourly or daily averages).
- **Sliding window algorithms:** These help efficiently compute metrics over moving time windows, such as rolling averages.

Data compression and retention policies

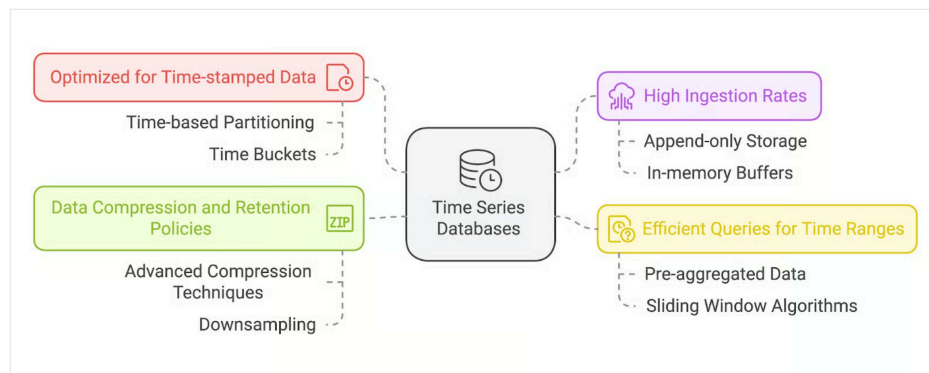
To manage the vast amount of time series data generated over time, TSDBs use advanced data compression techniques. These methods reduce storage requirements while preserving query performance.

TSDBs usually include retention policies so the users can define how long data should be kept. For example, a system might retain detailed data for the past month while *downsampling* for older data. Downsampling is the process of reducing the granularity of data over time. For example:

- Raw temperature readings might be recorded every 10 seconds for the most recent 7 days.
- For older data, the system might downsample to hourly averages to save space while still retaining historical trends.

Examples of advanced compression techniques include:

- **Delta encoding:** Storing the difference between consecutive values instead of the full value.
- **Gorilla compression:** A method used to efficiently compress floating-point time series data by storing changes in binary format.



Use Cases for Time Series Databases

Time series databases are used in many modern data-driven applications and across diverse industries. Let's explore the main use cases.

1. Internet of Things (IoT)

IoT devices, like smart thermostats, industrial sensors, and environmental monitors, generate continuous streams of time-stamped data. TSDBs are used to store and analyze this data, and power applications like:

- **Smart homes:** Monitoring and controlling appliances based on time-sensitive data.
- **Industrial automation:** Tracking machine performance and detecting anomalies in real time to minimize downtime.
- **Environmental monitoring:** Collecting data from sensors to track air quality, weather patterns, or water levels over time.

2. DevOps and system monitoring

In DevOps, TSDBs are widely used to monitor IT infrastructure and applications by collecting metrics like CPU usage, memory consumption, and network throughput. They enable:

- **Performance monitoring:** Visualizing system health and performance metrics in real time.
- **Anomaly detection:** Identifying unusual patterns, such as spikes in server load or network latency.
- **Capacity planning:** Using historical trends to predict and allocate future resource needs.

Tools like Prometheus and [Grafana](#) often integrate with TSDBs to provide visualization and alerting capabilities for DevOps teams.

3. Financial markets

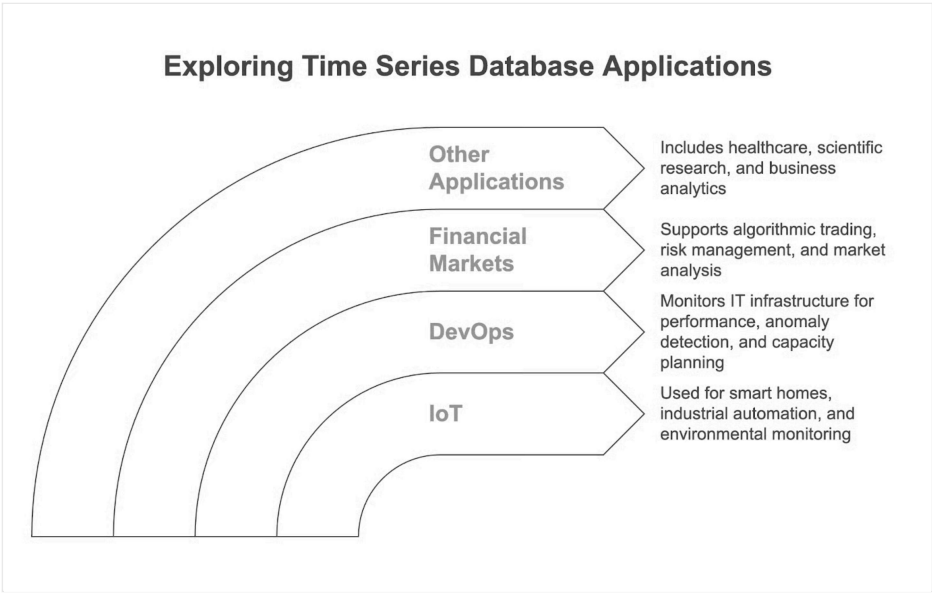
TSDBs are critical for processing and analyzing the vast amounts of high-frequency data generated in financial markets. They are used for:

- **Algorithmic trading:** Storing and analyzing market data in milliseconds to execute trades based on real-time trends.
- **Risk management:** Monitoring financial metrics over time to assess and mitigate risks.
- **Market analysis:** Analyzing historical data to identify patterns, trends, and anomalies in market behavior.

4. Other applications

While the three use cases above are very common, time series databases can also find applications in a variety of other fields:

- **Healthcare:** Monitoring patient vitals in real time and analyzing medical device data.
- **Scientific research:** Collecting and analyzing data for climate modeling, astronomical observations, and other time-dependent phenomena.
- **Business analytics:** Tracking customer behavior, [analyzing sales trends](#), and monitoring key performance indicators over time.



Best Time Series Databases: A Comparative Overview

Time series databases come in various shapes and forms, each tailored for specific use cases.

InfluxDB

InfluxDB is a popular open-source time series database developed by InfluxData. It was designed specifically for high ingestion rates and efficient querying of time-stamped data, making it a common solution for IoT monitoring, DevOps metrics, and real-time analytics.

Pros	Cons
High ingestion rates for massive volumes of data.	Requires manual management of retention policies for optimal storage.

SQL-like InfluxQL simplifies querying for analysts familiar with relational databases.	Scalability challenges for very large datasets without enterprise features.
Integrates easily with tools like Grafana for visualization.	Limited advanced query capabilities compared to SQL-based databases.

TimescaleDB

TimescaleDB is an open-source extension for [PostgreSQL](#), designed to combine the power of [relational databases](#) with time-series functionality. It allows you to leverage SQL while efficiently handling time-stamped data. This makes it particularly well-suited for use cases that require integrating time-series data with relational data, such as business analytics or IoT telemetry.

Pros	Cons
Full SQL support enables easy integration with existing PostgreSQL tools and workflows.	Requires PostgreSQL knowledge for setup and maintenance.
Hypertables: Automatically partition time-series data for efficient storage and queries.	May not yet match the ingestion speed of dedicated TSDBs like InfluxDB.
Combines relational and time-series data in a single database.	

Prometheus

Prometheus is a monitoring and alerting system with a built-in TSDB, widely adopted in DevOps for real-time system metrics, performance tracking, and alert management.

Pros	Cons
Lightweight and easy to deploy, especially with Kubernetes.	Limited long-term storage without external solutions.
Pull-based metric scraping ensures only relevant data is collected.	Scalability relies on additional tools like Thanos or Cortex.
PromQL provides powerful query capabilities.	Focuses on metrics and may not suit all general-purpose TSDB needs.

Clickhouse

ClickHouse is an open-source columnar database designed for high-performance analytical queries. While it is not a traditional TSDB, its architecture makes it exceptionally suited for [time-series data](#), especially when fast query performance is critical.

Pros	Cons
High query performance for analytical workloads.	Complex to set up and maintain for beginners.
Columnar storage reduces query latency.	Not specifically designed as a TSDB (may require workarounds).

Apache Cassandra

Apache Cassandra is a distributed NoSQL database built for horizontal scalability and high availability. While not exclusively a TSDB, it can be used effectively for time-series workloads, particularly when durability and fault tolerance are critical.

Pros	Cons
Excellent horizontal scalability.	Querying time-series data can be cumbersome without additional optimizations, as the database lacks native time-series query and aggregation features.
Fault-tolerant and highly available.	

Amazon Timestream

Amazon Timestream is a fully managed time series database service offered by AWS. Built for scalability and simplicity, it is ideal for organizations already leveraging AWS infrastructure for IoT and application monitoring.

Pros	Cons
Serverless architecture simplifies management.	Limited functionality outside the AWS ecosystem.
Scales automatically to handle large data volumes.	Costs can escalate with high data ingestion rates.

Let's do a quick recap with this table:

Database	Primary Use Cases	Storage Model	Ecosystem Compatibility
InfluxDB	IoT monitoring, DevOps metrics, real-time analytics	Time-series optimized, disk-based	Integrates with Grafana, Telegraf, and Kapacitor

TimescaleDB	Business analytics, IoT telemetry, SQL-based applications	Relational (PostgreSQL extension)	Works with PostgreSQL tools like pgAdmin, Grafana
Prometheus	DevOps monitoring, performance tracking, alerting	Embedded time-series storage	Designed for Kubernetes, integrates with Thanos, Grafana
ClickHouse	High-performance analytical workloads, time-series analytics	Columnar storage	Supports SQL-based analytics, integrates with BI tools
Apache Cassandra	Distributed time-series storage, large-scale logging	Wide-column NoSQL	Compatible with Spark, Kafka, and scalable distributed systems
Amazon Timestream	AWS-based IoT monitoring, application telemetry	Serverless, managed TSDB	Tightly integrated with AWS services (Lambda, IoT Core, QuickSight)

AWS Cloud Practitioner

Learn to optimize AWS services for cost efficiency and performance.

Learn AWS

Choosing the Right Time Series Database: Key Considerations

Time series databases offer different features to suit different needs. So, how do you choose the one that will best solve your problem?

Evaluate the data volume and ingestion rate

Time series workloads can vary significantly in terms of data volume (how much data is generated) and ingestion rate (how fast data is written). Some systems generate data sporadically, while others produce vast amounts of data every second.

For example:

- An IoT deployment with thousands of devices may require ingesting millions of data points every minute.
- A DevOps monitoring tool might collect real-time server metrics every few seconds across thousands of servers.

Not all TSDBs are optimized for extremely high write speeds. Systems that can't keep up with ingestion will drop data, leading to gaps in analysis and incomplete results.

If your systems generate a huge amount of data, you need to look for:

- TSDBs that support high ingestion rates without performance degradation.

- Solutions with append-only storage models and in-memory buffering for write optimization (e.g., InfluxDB, TimescaleDB, Prometheus).

I would really recommend that you evaluate the write throughput of a TSDB by testing it with realistic workloads. Benchmarking suites like [Time Series Benchmark Suite](#) can help simulate ingestion scenarios.

Alternatively, you can connect your system to the TSDB of your choice under a free trial and evaluate the results. Speaking from experience, realizing your pipelines generate 30GB of data every 30 minutes is not a fun surprise.

Look at query patterns and complexity

The type of queries you'll run frequently is another critical factor when choosing a TSDB. Different databases are optimized for different query workloads.

Examples of query patterns:

- **Simple range queries:** Fetching data points within a specific time range (e.g., "last 7 days").
- **Aggregation queries:** Calculating averages, sums, or percentiles (e.g., "average CPU usage per hour").
- **Downsampling:** Summarizing raw data into lower granularity (e.g., hourly averages from per-second readings).
- **Complex analytics:** Correlating multiple data streams, detecting anomalies, or running predictive analysis.

Some TSDBs excel at simple time-based queries, while others are better suited for complex aggregations or analytics workloads. Choosing the wrong database could result in slow queries and inefficient analytics pipelines.

You might want to look for:

- TSDBs with built-in support for your most common query types.
- Systems with SQL-like query languages (e.g., TimescaleDB with PostgreSQL compatibility) for more complex queries.
- Continuous query capabilities that allow you to precompute aggregations for frequently accessed data and reduce query load at runtime.

Check for scalability and availability

As your data volume grows, your TSDB must scale seamlessly while maintaining performance. Additionally, for mission-critical applications, the database must be highly available to ensure zero downtime.

If your TSDB can't scale horizontally (by adding more servers) or vertically (by increasing server resources), you risk system bottlenecks as data grows.

You want to look for:

- Horizontal scalability: Databases that can distribute data across multiple nodes (with [sharding](#) strategies, for example)
- High availability: Built-in clustering, replication, and failover mechanisms to ensure uptime.

Integration and ecosystem really matter

The ability of a TSDB to integrate with your existing tools and workflows is super important to ensure your team will adopt it and use it efficiently.

Look for:

- Integration with your tools (monitoring platforms, data processing framework, BI Tools, etc.)

- **Support for ingestion pipelines:** Look for TSDBs that integrate with your data sources, such as IoT devices, APIs, or log aggregators.
- **APIs and Query Languages:** TSDBs that offer REST APIs, SQL support, or language SDKs make integration easier for developers.

Cost

TSDB can get expensive, especially as your data volumes grow. Costs can vary depending on licensing, infrastructure requirements, and maintenance efforts.

Some TSDBs are open-source and free to use but may require significant infrastructure and operational costs. Others are commercial with licensing fees but come with advanced features and support.

Make sure you look at:

- **Licensing fees:** Open-source (e.g., Prometheus, VictoriaMetrics) vs. commercial (e.g., InfluxDB Enterprise).
- **Infrastructure costs:** Cloud-hosted TSDBs vs. self-hosted solutions.
- **Maintenance overhead:** Operational costs for scaling, backups, and disaster recovery.

For smaller workloads, you can consider cloud-hosted managed services that reduce operational overhead, but make sure you account for the long-term costs as data scales. I would also really recommend implementing data retention policies and downsampling from the start to manage storage costs effectively.

Conclusion

I hope you enjoyed reading this guide as much as I enjoyed writing it!

Now that you understand the basics of time series databases, you can pick a database of your choice, get a free trial, and start putting your skills into practice!

FAQs

How do you handle time series data with irregular intervals or different time zones? ^

For irregular intervals, resampling the data at regular time intervals can help, while time zone differences can be managed by converting all data to a single time zone or using UTC to maintain consistency.

Can I store time series data in a non-relational database? v

How do I choose between using a relational database or a specialized time series database for my project? v



AUTHOR

Marie Fayard

I am a product-minded tech lead who specialises in growing early-stage startups from first prototype to product-market fit and beyond. I am endlessly curious about how people use technology, and I love working closely with founders and cross-functional teams to bring

bold ideas to life. When I'm not building products, I'm chasing inspiration in new corners of the world or blowing off steam at the yoga studio.

TOPICS

Data Engineering



Training more people?

Get your team access to the full DataCamp for business platform.

For Business

For a bespoke solution [book a demo](#).

Learn data engineering with these courses!

TRACK

Associate Data Engineer in SQL

30 hr

Learn the fundamentals of data engineering: database design and data warehousing, working with technologies including PostgreSQL and Snowflake!

[See Details](#) →[Start Course](#)[See More](#) →

Related



BLOG

Database vs. Spreadsheet:
Comparing Features and...



BLOG

What is A Graph Database? A
Beginner's Guide



BLOG

Data Modeling Explained:
Techniques, Examples, and Bes...

[See More](#) →

Grow your data skills with DataCamp for Mobile

Qomplx LLC, Exhibit 2025

Make progress on the go with our mobile courses and daily 5-minute coding challenges.



LEARN

[Learn Python](#)

[Learn AI](#)

[Learn Power BI](#)

[Learn Data Engineering](#)

[Assessments](#)

[Career Tracks](#)

[Skill Tracks](#)

[Courses](#)

[Data Science Roadmap](#)

DATA COURSES

[Python Courses](#)

[R Courses](#)

[SQL Courses](#)

[Power BI Courses](#)

[Tableau Courses](#)

[Alteryx Courses](#)

[Azure Courses](#)

[AWS Courses](#)

[Google Sheets Courses](#)

[Excel Courses](#)

[AI Courses](#)

[Data Analysis Courses](#)

[Data Visualization Courses](#)

[Machine Learning Courses](#)

[Data Engineering Courses](#)

[Probability & Statistics Courses](#)

DATALAB

[Get Started](#)

[Pricing](#)[Security](#)[Documentation](#)

CERTIFICATION

[Certifications](#)[Data Scientist](#)[Data Analyst](#)[Data Engineer](#)[SQL Associate](#)[Power BI Data Analyst](#)[Tableau Certified Data Analyst](#)[Azure Fundamentals](#)[AI Fundamentals](#)

RESOURCES

[Resource Center](#)[Upcoming Events](#)[Blog](#)[Code-Alongs](#)[Tutorials](#)[Docs](#)[Open Source](#)[RDocumentation](#)[Book a Demo with DataCamp for Business](#)[Data Portfolio](#)

PLANS

[Pricing](#)[For Students](#)[For Business](#)[For Universities](#)[Discounts, Promos & Sales](#)[Expense DataCamp](#)[DataCamp Donates](#)

FOR BUSINESS

[Business Pricing](#)[Teams Plan](#)[Data & AI Unlimited Plan](#)[Customer Stories](#)[Partner Program](#)

ABOUT

[About Us](#)[Learner Stories](#)[Careers](#)[Become an Instructor](#)[Press](#)[Leadership](#)[Contact Us](#)[DataCamp Español](#)[DataCamp Português](#)[DataCamp Deutsch](#)[DataCamp Français](#)

SUPPORT

[Help Center](#)[Become an Affiliate](#)[Privacy Policy](#)[Cookie Notice](#)[Do Not Sell My Personal Information](#)[Accessibility](#)[Security](#)[Terms of Use](#)

© 2026 DataCamp, Inc. All Rights Reserved.