Time-Series Databases

Definition: a database optimized for time-stamped (time series) data and for measuring change over time. [1]

Data is timestamped, a.k.a. time-series data. [2]

Overview:
A time series database is usually suited to the characteristics of timestamped data, this usually consists of being 'time-centric, recent, and normally append-only' [2]

Time series data scales quickly which automated relational databases struggle with.

Time Series Databases treat a time element as 'a first class citizen' [2] (Key element?)

NoSQL is seen as the best for time series, but a larger number of alternatives can be adapted to the situation (e.g. PostgreSQL) [2]

Time Series Examples:
Sensor data
Transaction data / Financial transactions / Customer transactions / Order history
Operational analytics / Application data
Fleet data /Logistics
Metrics data
Tick data / Fintech data / Trading data
Event data
Vector data
Weather data
Insurance data
Call records

'If you want to know if your data is time-series data, this is the litmus test: does your data have some kind of timestamp or time element related to it, even if that may not be its main dimension? If the answer is "yes," you're dealing with time-series data.' [2]

Benefits: [2]
Increase efficiency of ingesting or querying data
Implement shortcuts
internal optimizations like auto-partitioning and indexing
specialized features to simplify and speed up the calculation of typical time-series queries
Data management (e.g. deleting data older than a year)
Time based aggregation

Cons:
May scale to quickly (become unwieldy and difficult to manage)
Hyper-specialised
Context may be lost due to the data being handled based on time rather than content

Examples of Time-Series Databases: [2] [3]
InfluxDB – Scalable, difficult to work with
Kdb – Handles large data sets well, high cost associated (kdb+?)
Prometheus – Simplicity, scalable, fault tolerance, limited community support
DolphinDB – Scalibility, performance, high cost
Apache Druid – Real time analytics, complex setup
MySQL – Versatile, fast, efficiency decreases with size
PostgreSQL – Flexible, large number of features, lacks comparison features

Key aspects: [2]
Scalability – can handle vertical and horizontal scaling.
Maintainability – automation,
Reliability
Query Languages – some have custom specific languages to learn.

Key Takeaways:
Horizontal and vertical scalability
Query language
Automation
Data management (e.g. deleting data older than a year)
Time based aggregation
Hyper-specialised

(Personal input: MySQL is referenced a lot and seems to be the one most utilised)

References:
[1] InfluxData. (2022). *Time Series Database (TSDB) Guide | InfluxDB*. [online] Available at: https://www.influxdata.com/time-series-database/#:~:text=Here%E2%80%99s%20a%20brief%20time%20series%20database%20definition%3A%20A [Accessed 15 Feb. 2024].

[2] Timescale Blog. (2023). *Time-Series Database: An Explainer*. [online] Available at: https://www.timescale.com/blog/what-is-a-time-series-database/#:~:text=A%20few%20examples%20of%20specific%20time-series%20data%20types%3A [Accessed 15 Feb. 2024].

[3] InfluxData. (2022). *Time Series Database (TSDB) Guide | InfluxDB*. [online] Available at: https://www.influxdata.com/time-series-database/#:~:text=Here%E2%80%99s%20a%20brief%20time%20series%20database%20definition%3A%20A.