Why do we need a time-series database

With the development of IoT, 5G and other technologies, the demand for sequential data in industrial Internet of Things, smart home, monitoring and other industries shows explosive growth, while traditional relational database is difficult to effectively deal with this. So what is a time-series database.

To understand what a time series database is you have to understand time series data. Time series data is the data collected at different times and used to describe the changes of phenomena over time. This type of data reflects the state or extent to which something, phenomenon has changed over time. These data sets have three main things in common:  Incoming data is almost always recorded as a new entry；Data usually arrives in chronological order  ；Time is a principal axis (regular or irregular time interval)

In other words, the processing of time series data is usually carried out with the arrival of data.  Although incorrect data needs to be corrected after the fact, or delayed or unordered data needs to be processed, these are exceptions and not standards. You may ask: How is this different from adding fields to a database? So, it depends: How does your data set track change?  Update the current entry or insert a new one? When you collect a new reading, do you overwrite the previous reading, or do you create an entirely new reading on a new line?  Although both methods can provide you with the current state of the system, only the second method can keep track of all the states of the system. In short: time series datasets track changes to the entire system and insert new data rather than updating existing data.

 What makes time series data so powerful is that each change in the system is recorded as a new row, allowing you to measure change: analyze changes in the past, monitor changes in the present, and predict how they will change in the future.

 Therefore, we define time series data as data that uniformly represents changes in a system, process, or behavior over time.

A time-series database (TSDB) is a computer system that is designed to store and retrieve data records that are part of a “time series,” which is a set of data points that are associated with timestamps. The timestamps provide a critical context for each of the data points in how they are related to others. Time series data is often a continuous flow of data like measurements from sensors and intraday stock prices. A time-series database lets you store large volumes of timestamped data in a format that allows fast insertion and fast retrieval to support complex analysis on that data.

Some people may wonder why we can't use traditional database to store these time series data like PostgreSQL or MySQL. Many of our users started off working with time series by storing their data in common SQL RDBMSes like PostgreSQL or MySQL. Generally they find this works for a time, but things start to fall apart as the scale of the data increases.

Create a single table to store everything with the series name, the value, and a time. Separate lookup index if we wanted to search on anything other than the specific name (like server, metric, service, etc.).

This naive implementation would have a table that gets A huge amount of new records per day. This would quickly cause a problem because of the sheer size of the table. With time series, it's common to have high-precision data that is kept around only for a short period of time.

This means that soon you'll be doing just as many deletes as inserts, which isn't something a traditional DB is designed to handle well. Create a separate table per day or some other period of time. Requires the developer to create application code to tie the data from the different tables together. More code must be written to compute summary statistics for lower-precision data and to periodically drop old tables. Then there's the issue of scaling past what a single SQL server can handle. Sharding segments of the time series to different servers is a common technique but requires more application-level code to handle it. In a word, relational technologies were not designed to solve the specific time series issues, and trying to get them to solve them is impractical.

So what are the essential differences between sequential databases and traditional big data storage solutions?

I think the most important difference is structured data.

Structured data is stored. We all know that the traditional big data solution to store the data contains structured, semi-structured, and unstructured data, thus decided the we can't decide what field and define the field of data types, like hbase is unified storage by byte type, that is to say on the hbase is the data in the byte array, Converting from a normal type to a byte array is something we have to do ourselves, and we don't know how to convert to byte to make it more efficient for storage. However, the data generated by sequential data are all structured data. We can define the fields and types of data in advance, and let the database system select the optimal compression mode according to different field types, thus greatly improving the storage utilization rate. [at](javascript:;) [the](javascript:;) [same](javascript:;) [time](javascript:;), analysis aggregates structured data. Since analysis aggregations are structured data, we don't need to use complex computing tools like MapReduce, and generally don't need data warehouses like Hive. Instead, we just need to consolidate the database storage level with computing tools like SUM and AVG, and we can even do some simple streaming calculations. It provides the basis for "super fusion" (super fusion means that multiple components similar to the previous big data processing scheme are fused into one component, mainly because structured data is too simple, collection and calculation are relatively simple, which is also the development trend of sequential database in the future to reduce the system complexity).

In these respects, the biggest characteristics of the two sequential databases are scale and availability. Scale: Time series data is accumulated very quickly. (For example, a connected car can collect 25 gigabytes of data per hour.) Conventional databases are not designed to handle data of this size, and relational databases are very poor at handling large data sets; The NoSQL database handles scale data well, but not as well as a database fine-tuned for time series data. In contrast, time series databases (which can be based on relational or NoSQL databases) treat time as a first-class citizen, processing such large data volumes with increased efficiency and performance gains, including: Higher Ingest Rates, faster large-scale queries (although some support more queries than others), and better data compression.

For developers, it can bring great convenience in the development process.

Firstly, we could optimize the database around some of the peculiarities of time series. It’s insert only, we need to aggregate and down sample, we need to automatically evict high-precision data in the cases where users want to free up space. We could also build compression that was optimized for time series data. We also organized the data in a way that would index tag data for efficient queries. At the database level, there were many optimizations we could get. The other advantage in building specifically for time series is that we could go beyond the database. We’ve found that most users run into a common set of problems they need to solve—how to collect the data, how to store it, how to process and monitor it, and how to visualize it.

This is why developers are increasingly adopting time series databases and using them for a variety of usage scenarios:  Monitoring software systems: VMS, containers, services, and applications; Monitor physical systems: devices, machines, connected devices, the environment, our houses, our bodies; Asset tracking applications: Cars, trucks, physical containers, Pallets; Financial trading systems: traditional securities, emerging cryptocurrencies; Event applications: Track user and customer interaction data; Business intelligence tools: Track key metrics and the overall health of the business; And more scenes.

All of the above illustrates the importance of sequential databases. In the Internet of Things, as data volumes explode, more and more scenarios exceed 500,000 or even 10 million measurement points per second.  For this level of data, the stand-alone sequential database can no longer meet the business requirements.  At the same time, in recent years, distributed timing database technology has become more and more mature and perfect, and has entered the stage of large-scale application.  Although the application of time series database in Internet of Things is still in a relatively shallow stage, enterprises gradually realize the value of data, and more and more enterprises hope to mine more valuable information by using time series database in the future. Therefore, the development of time series database is the general trend of the future. With time series database, we will develop more scientific hotspots to change people's lives.