**Why do we need a time-series database**

Before introducing time-series database, we need to know what time-series data is. Time-series data is a series of data based on time. Connect these data points into lines in time-dependent coordinates, and in the past, multi-latitude reports can be made to reveal their trends, regularities, and anomalies; In the future, big data analysis and machine learning can be used to achieve prediction and Warning.

So, a time-series database is a database that stores time-series data, and needs to support basic functions such as fast writing, persistence, and multi-dimensional aggregation query of time series data. It has the following main features：

·Data characteristics: The amount of data is large, the data grows with time, the same dimension repeatedly takes values, and the indicators change smoothly (the track coordinates of a vehicle’s smooth change uploaded by a certain device).

·Writing features: high concurrent writing, and will not be updated (the trajectory will not be updated).

·Statistical analysis is performed on indicators according to different dimensions. There are obvious hot and cold data. Generally, only recent data will be queried.

So what are the essential differences between time series 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 data to be stored in traditional big data solutions includes structured, semi-structured, and unstructured data, which determines that we cannot decide which fields and the data types that define each field, such as hbase through the byte type. Unified storage, that is to say, the data placed in hbase are all byte arrays. We need to do it ourselves to convert from ordinary types to byte arrays. We don't know how to convert them into byte arrays, and their storage efficiency will be higher. However, the data generated by time series data is structured data. We can define the fields and types of the data in advance, and let the database system choose the optimal compression method according to different field types, which greatly improves the storage utilization.

Let's observe a real life example：the average temperature for several consecutive days at one location. For the past few decades, average air temperature has been used as the main reference factor for building energy efficiency. During any given week, the daily average temperature in the same location may vary only slightly, but at the same time, factors affecting the environment may have changed dramatically. Instead, knowing the temperature changes for each hour of the day, combined with the amount of precipitation, cloud cover, and wind speed during that time, can greatly improve a property's ability to model and optimize energy efficiency.

Another example is to keep up with the current facts, we want to record various data of new coronary pneumonia patients in a certain area. Undoubtedly, these data require us to keep track and update at any time. Such as patient age, admission or discharge, days of recovery, etc., beneficial time series databases help us understand how daily statistics are derived, allowing us to better analyze trends, accurately report totals, and take action, and even influence the government policy.

Through the above real cases, we can summarize three common points of time series databases:

1. The data written is almost new.

2. Data is usually written in chronological order.

3. Time is a main axis.(time intervals can be regular or irregular)

In other words, the way time series data is written is usually "append". While data corrections, processing delays, out-of-order data, etc. may need to be done after the fact, these are anomalies, not the norm.

This makes TSDB the fastest growing database category today for two reasons: scale and availability：

Scale: Time series data accumulates very quickly, and normal databases are designed to handle this scale (at least not in an automated way). Relational databases perform poorly on very large datasets, while NoSQL databases perform better at scale (although relational databases fine-tuned for time series data can actually perform better, as we benchmark against InfluxDB, Cassandra and MongoDB compared to that shown). In contrast, the benefits introduced by a time series database (whether relational or nosql-based) are only possible if you put time as your first consideration. These benefits allow them to deliver massive performance improvements, including higher throughput and faster large-scale queries, as well as better data compression.

Availability: TSDB usually also includes built-in functions and operations commonly used for time series data analysis, such as data retention policies, continuous query, flexible time aggregation, etc. Even if you're just starting to collect this type of data and don't need to think about scale just yet, these features can still provide a better user experience and make data analysis tasks easier. Using built-in functions and features to analyze readily available trends in your data layer often finds unexpected value, no matter how large or small your data set is.

Today, we can see that time series databases are used in all walks of life：

1. Smart city and energy industry intelligent emergency command and integrated communication dispatch. Focusing on monitoring, command, dispatch, conference, communication and other functions. . Real-time and effective visual command is realized in all aspects of emergency warning, reporting, response, command, etc., to meet the rapid response needs of real-time image transmission and video consultation on the scene of emergencies.
2. Intelligent inspection and security of the park. There are many application scenarios in daily inspection of various parks, reporting of hidden dangers, 3D maps and integrated scheduling. Equipment management operation status, HSE risk level, process flow, process control operation parameters and other maintenance conditions Visual display of various business sites and management real-time data and information, timely detection of problems, analysis of the reasons, put forward rectification suggestions, and implement them.
3. Intelligent operation and maintenance of equipment in the energy industry. The IoT platform can also be applied to the unified management and operation and maintenance of massive equipment terminals, online monitoring and diagnosis of equipment status, and timely fault warning. Operation and maintenance data can also be displayed through multi-dimensional charts.

Once time-series data is used to store more information, we still have to choose the data model, read-write mode, and time-series database that best suits the business. Although NoSQL time-series databases have prevailed over the past decade as the preferred storage medium, more and more developers see the disadvantage of storing time-series data separately from business data (most time-series databases do not provide as good a service as relational data). support). In fact, this poor developer experience is one of the main reasons we developed FastData For TSDB. Keeping all data in one system can greatly reduce application development time, as well as facilitate quick critical decisions.

With the rise of a large number of self-service business intelligence tools such as Tableau, Power BI, and even Excel, it is difficult for users to make timely and business-critical analysis and observations when valuable time series data is separated from business data. Instead, users find they need to rely on these third-party tools to analyze meaningful information from their data. There are many valid reasons to use these powerful tools.

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 past change, monitor present change, and predict how it will change in the future.

Say you're maintaining a web application, and every time a user logs in, you can update the user's "last\_login" timestamp in a row in the "users" table, but if you treat each login as a separate event, and What would be the effect of collecting this data over time? Then you can: track historical login activity, see how user usage has increased or decreased over time, differentiated users based on frequency of app visits or more.

And, when applied to this scenario, we can see the advantages of time series database in data storage and analysis：

Storage cost: Taking advantage of the characteristics of time increment, dimension repetition, and smooth change of indicators, the coding and compression algorithm is reasonably selected to improve the data compression ratio; historical data is aggregated through pre-reduction precision to save storage space.

High concurrent writes: Write data in batches to reduce network overhead; data is first written to memory, and then periodically dumped for immutable file storage.

Low query latency, high query concurrency: Optimize common query patterns, reduce query latency through technologies such as indexing, and improve query concurrency through technologies such as caching and routing.

We've noticed that many applications may never need time series data (and it works better with a "current state view"). But as the exponential curve of technological progress continues to move forward, it seems that these "current state views" become unnecessary. Only by storing more data in time series form will we be able to understand it better.

Over time, major cloud vendors have also launched their own time series databases. Alibaba's TSDB team has gradually served DBPaaS, Sunfire and other group businesses since the first version of the time series database was launched in 2016. After the public beta in mid-2017, it was officially commercialized at the end of March 2018. In terms of technology, TSDB has continuously absorbed the strengths of various companies in the time series field, and has opened up the development path of self-developed time-series databases.