Time series database helps the development of big data era

What is a sequential database? In a word, it is a database used to store time data. This time data is different from general data. It is time sequence data, namely time series data, which is recorded and indexed sequentially in time dimension. According to the statistical survey of the last two years, time series database has maintained the fastest growth. Why does this happen? Classical physics model system in the past, the time is constant in this dimension is not affected, the world of all things run what happened what time can be used to describe status, in time for the shaft can be described the development of things well, in modern times, with the development of science and technology, more and more high performance electronic products gradually onto the world stage, The computing power of a palm-sized computer today is far, far higher than that of a computer the size of a square in decades past. In a sense, today's computing devices can handle the computation and simulation of large amounts of data, which is why the rise of databases, so why do we need sequential databases? I think there are the following reasons.

First, the present many data in constant sampling, such as measurement of fermentation product concentration in biological engineering, measuring the local weather forecast rainfall and wind speed, smelting iron and steel measuring steel melt when temperature changes, these samples have a common over time, the steady output with the data of time, the automated way causes data is very huge, How to use an effective way to deal with this is a problem, the use of time database can undoubtedly be very good to complete sampling and recording, facilitate the formation of a good system to complete processing and reporting.

Secondly, the data sampling rate becomes a challenge. When the collected data is used as the coordinate axis with time, the higher the measured frequency at the same time, the graph composed by the lines of each sample point in the graph becomes smoother, and such data is more accurate for fitting the equation with mathematical methods. For example, compared with the data graph recorded in unit second, the data graph recorded in unit minute is like the relationship between line graph and curve in general the higher the data sampling rate is, the more accurate the final result will be to explore the connection behind, and this exploration method is also more in line with scientific exploration. The timing database itself is designed to address this challenge, which explains why it is needed.

Then as before, the database is very large, the recorded information when received how vast amounts of data query is often a challenge, if use traversal way to query a certain information, or modify a certain value, often at this time the time spent is very large, in other ways, such as binary tree search, although than traverse to save a lot of time, However, in this case, many nodes need to be traversed, and the sequential database approach is a good solution to this problem. The time parameter, like dictionary lookup, can be very fast to find the required data, which is why the sequential database is popular.

The second is the challenge of storage. For example, since data collection is a massive amount of monitoring data, it will take up a lot of storage space if the traditional database is used for storage. If we use a relational database to store 10,000 monitoring data, it will take up a lot of storage space. Even if index and other factors are not considered, 5-6T storage space is required, which does not include the storage time and other factors related to the monitoring point. If all of them are included, then index building will require 15T-20T storage space. Real-time database USES special compression algorithms, including Harvard, algorithm, revolving door algorithm and some secondary compression algorithm, compression ratio can reach around space generally, plus for special processing time and indexing, storage can shrink to 1/40 of the relationship between library, therefore, the above example only need 500 g of space can be effectively stored. This is a good example of why sequential databases are needed.

One might ask, why not use a common database?

The truth is you can, and some people already have. However, TSDB is the fastest growing database category today for at least two reasons: size and availability.

Scale: Temporal data accumulates very quickly, and regular databases are designed to handle this scale (at least not in an automated way). Relational databases perform poorly on very large data sets, while NoSQL databases perform better on scale (although relational databases fine-tuned for sequential data can perform better, as we showed in benchmark tests compared to Influx DB, Cassandra, and MongoDB). In contrast, sequential databases (whether relational or NoSQL-based) introduce benefits that are only possible if you consider time as a primary consideration. These benefits enable them to provide large-scale performance improvements, including higher throughput and faster large-scale queries, as well as better data compression.

Availability: TSDB also typically includes built-in functions and operations commonly used for temporal data analysis, such as data retention policies, continuous queries, flexible time aggregation, and so on. Even if you're just starting to collect this type of data and don't need to worry about scale right now, these features can still provide a better user experience and make the task of analyzing data easier. Using built-in functions and features to analyze ready-to-use trends in the data layer often reveals unexpected value, no matter how large or small your data set is.

Therefore, developers are increasingly adopting sequential databases and applying them to a variety of scenarios:

· Monitoring software systems: virtual machines, containers, services, applications, etc.

· Monitoring physical system: equipment, machinery, connecting equipment, environment, family, human body, etc.

· Asset tracking applications: vehicles, trucks, physical containers, pallets, etc.

· Financial trading systems: typical securities, nascent cryptocurrencies, etc.

· Event application: tracking user/customer interaction data, etc.

· Business intelligence tools: Track key metrics and overall business health, etc.

, etc.

Once the sequential data is used to store more information, we still must choose the sequential database that best fits the business data model, read, and write mode, and development technology line. Although NoSQL time series databases have gained popularity as the preferred storage medium over the past decade, more and more developers see the disadvantages of storing sequential data separately from business data (most sequential databases do not provide good support for relational data). In fact, this poor developer experience is one of the main reasons we developed Fast Data 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 self-service business intelligence tools like Tableau, Power BI, and even Excel, it's hard to make timely, business-critical analytics and observations when valuable sequential data is separated from business data. Instead, users find themselves relying on these third-party tools to analyze meaningful information from the data. There are many valid reasons to use these powerful tools, but the ability to quickly query timing data and meaningful metadata information should not be one of them. SQL has been tested for decades and provides a mature and effective way to generate these valuable aggregations and analyses.

The point is, knowing where temporal data is, and where to store it, will have a huge impact on future developments.

Here are some ways to look at it:

The first is the design of storage data structure, using the architecture of XLSM-Tree to solve the problem of high write throughput.

Second, on high-performance queries, each product has its own index customization and the introduction of query optimizers.

Third, in terms of storage cost, each sequential database product selects column storage and specific compression algorithms to solve the storage cost problem. Of course, in the cloud storage cost, we can also do more optimization at the edge of the processing, hot and cold data processing in the cloud, which is also the trend of distributed cloud technology strategy.

The fourth and very important problem in the field is the solution of out-of-order, which can be solved at the storage level by preserving the order before writing and rearranging the order after writing. And then play the greatest value in the subsequent calculation and analysis. So, think about any other key issues and technologies that need to be addressed in addition to those four core areas?

Of course, in the distributed cloud architecture, the deployment of edge side also needs to be highly reliable, and each sequential database product needs to provide multi-copy cluster edition support.

Finally, real-time computing at the edge is also necessary to maximize the value of a sequential database, and there are technical challenges to supporting real-time computing in a sequential database.

Front of temporal database can be applied to a variety of scenarios including various monitoring field, as well as the intelligent manufacturing, intelligent life, smart city and so on in the scene, so want to value maximization of these scenarios, we need to consider from the collected data analysis and data visualization of each link of different challenges. What I want to emphasize here is that the best way to maximize data is to establish a closed loop of data at the cloud side. We are not only collecting data and monitoring data, data visualization, but the maximum value of data business also needs to be analyzed on the collected data, and the analyzed data then reverse control terminal to achieve data closed loop.