Why do we need a time-series database?

For the first time ever, worldwide interest in time series data has peaked in the most unexpected way. The COVID-19 pandemic has turned billions of people around the world into consumers of time-series data and demanded accurate and timely understanding of daily COVID-19 trends. Compared to stock market trends, self-driving cars, and predicting the exact time of your next online purchase, there have been many examples in recent years of how time-series data collection and analysis can impact an individual's daily life. In our information-hungry world, access to detailed, feature-rich time-series data has become one of the most valuable commodities. Businesses, governments, schools and communities, large and small, are looking for ways to extract value from analyzing time-series data. The usage patterns of software developers already reflect the same trend. The time series database (tsdb) has been the fastest growing database for the past two years.

Time series data, that is, time series data, data that is recorded and indexed in the order of the time dimension. Various types of devices in fields such as smart cities, Internet of Things, Internet of Vehicles, and Industrial Internet will generate massive amounts of time series data. These data will account for more than 90% of the world's total data. It is convenient for us to analyze the data according to the change of time.

Time series data is everywhere

Suppose you maintain a web site. Every time a user logs in, update the user's "last\_login" timestamp in the "users" table. But what if you treat each login as a separate event and collect them over time? With this type of time series data, it is possible to analyze historical login activity to see how usage has changed over time, segment users based on how often they access the app, and more.

Another example is critical to every IT group around the world, operational metrics for servers, networks, applications, environments, etc. This timing metric data is critical to guaranteed service reliability. By tracking changes in each metric, IT can quickly identify problems, plan for upcoming events, and diagnose whether application updates have resulted in changes in user behavior, for better or worse. These examples illustrate a key point that preserving the inherent time-series properties of data allows us to retain valuable information, such as how data has changed over time. We note that both examples describe a common type of time series data - event data. Of course, there is an obvious problem with storing data in this way: you end up with a huge amount of data, and it grows very fast. So here's the problem: being able to analyze increased time series data is more valuable than ever, but it piles up very quickly. Massive amounts of data can cause a series of problems, whether it is storage, or fast querying, which is why people are more inclined to use time series databases than ever before. The world demands that we make faster and better data-driven decisions. Traditional static data cannot solve this problem. To meet demand, you need to collect data with the highest possible fidelity - that's what time series data provides: everything that happens in a system can be stored like a movie, whether it's software, physical power plants, games or applications customers in .

The time series database has the following characteristics:

1. 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).

2. Write characteristics: high concurrent write, and will not be updated (the trajectory will not be updated).

3. Query characteristics: Statistical analysis of indicators according to different dimensions, there is obvious hot and cold data, generally only recent data (generally we only care about recent trajectory data).

There are essential differences between time series databases and traditional big data storage solutions:

1. It stores structured data. 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.

2. Analysis and aggregation is structured data. Since the analysis and aggregation is structured data, then we do not need to use complex computing tools such as mapreduce, nor do we generally need data warehouses such as hive, but only need to cohere at the database storage level, similar to sum and avg in this calculation. The tools are enough, and even some simple stream computing can be done, which provides the basis for 'hyper-convergence' (hyper-convergence means that multiple components similar to the previous big data processing solutions are merged into one component, mainly because Structured data is too simple, and collection and calculation are relatively simple, which is also the development trend of subsequent time series databases, reducing system complexity).

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 in 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.

This is why developers are increasingly adopting time-series databases and applying them in a variety of scenarios:

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

Monitor physical systems: equipment, machinery, connected devices, environments, homes, humans, etc.

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

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

Event application: track user/customer interaction data, etc.

Business intelligence tools: track key metrics and the overall health of the business, etc.

· and many more

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.

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, when valuable time series data is separated from business data, it is difficult for users to make timely and business-critical analysis and observations. 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, but being able to quickly query time series data and meaningful metadata information should not be one of them. SQL has been tried and tested for decades and provides fairly mature and efficient ways to generate these valuable aggregations and analyses.

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