Let's first discuss what is time series data and its character

Time series data, namely time series data, is recorded and indexed sequentially according to the time dimension. Various types of devices in areas such as smart cities, Internet of Things, Internet of vehicles and industrial Internet will generate massive amounts of temporal data. These will account for more than 90% of the world's data. It is convenient for us to analyze the data according to the time change. Previously, our view of time series data was static: daily temperature highs and lows, stock market opening and closing prices, even daily or cumulative hospitalizations due to COVID-19. However, we tend to overlook the potential for subtle differences in these static data. For example: If I give you 10 yuan, the bank's traditional database will have a charge to my account and a charge to your account. And then, if you give me 10 yuan, the same process goes the other way. At the end of the day, our bank balance still looks the same, and as far as banks are concerned, nothing has changed this month. But with a sequential database, the bank could sense that there might be a deeper reason why these two people keep exchanging $10 notes. If you track this slight discrepancy, your month-end account balance takes on more meaning. Or the average temperature over several days in a row in one location. Over the past few decades, average temperature has been used as the main reference factor for building energy efficiency. In any given week, the average daily temperature in the same place may be only slightly different, but the factors that affect the environment can change dramatically over the same period of time. Conversely, knowing the hourly temperature changes during the day, coupled with the amount of precipitation, cloud cover and wind speed during that time, can greatly improve the ability to model properties and optimize energy efficiency.Similarly, while it is valuable to know the total number of daily COVID-19 hospitalizations in a community, that number alone does not provide a good picture of the details. For example, a hospital might disclose that 20 people were admitted on Monday and that the number increased slightly throughout the week, to a total of 23 by Friday. At first glance, there was a 15% increase in hospitalizations this week. But if we count the detailed records (increasing the frequency of collection), we might see a net increase of 3 cases this week, but in fact, 10 people have been discharged, 13 have been added, and in the last 5 days, there has been a 65% increase in new admissions. Tracking various aspects of patient data over time (such as patient age, admission or discharge, days of recovery, etc.) helps us understand how to produce daily statistics, enabling us to better analyze trends, accurately report totals, and take action that may even influence government policy.

These examples illustrate how different modern time series data are from what we used to know. Time series data analysis is much deeper than pie charts or Excel. The data is not just a measure of time, the point is to help us analyze the data and get valuable information.

So what are the essential differences between sequential databases and traditional big data storage solutions? I have some reason below

First of all, it stores structured data. As we all know, traditional big data solutions store structured, semi-structured and unstructured data, which determines that we cannot determine the fields and define the data types of each field. For example, hbase is uniformly stored by byte type. In other words, data stored in hbase is byte array. We need to convert data from common types to byte array by ourselves. We do not know how to convert data to byte to improve storage efficiency. 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.

Second, 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", which means to fuse multiple components similar to the previous big data processing scheme 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, reducing the system complexity.

So why use a sequential database? Here I summarize the following reasons

The first is scale: temporal data accumulates very quickly, and regular databases are designed to handle this scale. 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 actually perform better, as we showed in benchmark tests compared to InfluxDB, Cassandra, and MongoDB. In contrast, the benefits introduced by a sequential database are only possible if you consider time as the 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.

Second is 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.

Moreover, in real life, temporal data is everywhere. Here are a few examples: Suppose you maintain a Web site. Update the last\_login timestamp of the user in the Users table each time the user logs in. But what if you treated each login as a separate event and collected them over time? With this kind of sequential data, you can analyze historical login activity, see how usage changes over time, break down users based on how often they visit the application, and so on. Another example is critical to every IT group around the world, the operational metrics of servers, networks, applications, environments, etc. Such timing metrics are critical to ensuring 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. The temporal nature of preserving data allows us to retain valuable information, such as how the data has changed over time. Notice that both examples describe a common temporal data type, event data. Of course, there's an obvious problem with storing data this way: You end up with huge amounts of data, and they grow very fast. So here's the problem: being able to analyze increased temporal data is more valuable than ever, but it's piling up really fast. Large amounts of data can cause a number of problems, from storage to quick queries, which is why people are more inclined to use sequential databases than ever before. The world demands that we make data-driven decisions faster and better. Traditional static data cannot solve this problem. To meet your requirements, you need to collect data with the highest fidelity possible -- this is what sequential data provides: everything that happens in the system can be stored like a movie, whether it's software, a physical power plant, a game, or a customer in an application.

Below is the reason why developers are increasingly using sequential databases for a variety of scenarios.

I find Several examples and list them below. The first is intelligent emergency command and integrated communication scheduling for smart cities and energy industry. It adopts digital BIM+GIS+NBIOT+AI+5G+ algorithm technology to integrate multiple functions such as monitoring, command, scheduling, conference and communication. In the emergency warning, reporting, response, command and other aspects of the realization of timely and effective visual command, to meet the emergency scene real-time image transmission and video consultation requirements for rapid response. Then, the park intelligent inspection and security, equipment management and operation status, HSE risk level, process flow, process control and operation parameters and other maintenance conditions of all kinds of on-site business and management real-time data and information are displayed visually. Problems are found in time, causes are analyzed, rectification suggestions are put forward and implemented. Finally, intelligent inspection and security, comprehensive monitoring of combustible gas, smoke, electrical fire; Full link linkage of fire sensing, video monitoring and fire water; The whole process of fire, alarm, evacuation and fire fighting is covered