Why do we need a time series database

**Abstract:**

With the advent of the Internet of Things, more and more data needs to be indexed chronologically. Like vehicle trajectory data, sensor temperature data. Therefore, with the development of the Internet of Things, the amount of time data will greatly increase, so the development and application of time sequence database becomes very important. This article focuses on what time series data is, the scenarios of using time series data, the characteristics of time series database, and the problems that time series database solves, and discusses why we need time series database.

**Key words:**

Time series data database

**1. What is time series data**

**1.1 Definition of time series Data:**

Temporal data, by definition, is a string of data indexed by time dimension. Use descriptive language to explain what time series data is. Simply put, this kind of data describes the measured value of a subject at each point in a time range. It describes the changing information of an object's state in the historical dimension. With the development of Internet of Things, big data and artificial intelligence technology, time series data is also showing an explosive growth.

**1.2 Characteristics of time series Data:**

For temporal data, it has several notable features. 1. Data features: There is a large amount of data, the data increases over time, the value of the same dimension is repeated, and the indicator changes smoothly. 2. Write features: High concurrency and no update. 3. Query features: Statistical analysis is performed on indicators in different dimensions, and there are obvious hot and cold data. Generally, only recent data will be queried.

**2. Scenarios using sequential databases**

The large amount of data and the change of data over time are one of the main characteristics of sequential data, and there are many application scenarios based on these characteristics. The main scenarios using sequential data include virtual machines, intelligent transportation, smart homes, sensors and so on. Trading systems in the financial sector: typical securities, nascent cryptocurrencies, etc. In addition to monitoring the physical system: equipment, machinery, connected devices, environment, home, human body are all using time series data, in these application scenarios can consider using time series database. Therefore, from the perspective of the scene of generating temporal data, there will be such data generation scene in every aspect of social life. This explains why we need to develop sequential databases.

**3. Characteristics of time series database**

The first characteristic of a sequential database is that it can process a relatively large amount of sequential data. Temporal data accumulates very quickly, and normal databases are not designed to handle this scale, at least not in an automated way. Relational databases perform poorly on very large data sets. 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.

The second characteristic of sequential database is high availability. Temporal databases also typically include built-in functions and operations commonly used for temporal data analysis, such as data retention policies, continuous queries, flexible temporal 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 the data set.

Time series data need to be focused on rapid intake. That is, new data is always inserted. For the most part, these are add-on operations, adding only the most recent time series data -- although users sometimes need to backfill history, and we often see lagging data collection in the sensor data use case. Even with the latter, you usually add the most recent data to each individual series. High precision data has a short storage time, while mid - and low-precision summary data has a long storage time. One way to think about this is with the original high-precision samples and summaries of the 5 minute and 1 hour intervals. In practical terms, this means that you must constantly delete data from the database. High-precision data resides for a short time and should then be purged. This is a different workload than the normal database design handles. The query pattern for time series is very different from other database workloads. In most cases, the query will pull back a piece of data for the requested time range. For databases that can dynamically compute aggregation , they will frequently go through many records to return result sets for queries. It is critical to iterate quickly over many records to calculate the aggregation for a time series use case.

**4. Advantages of time series database**

**4.1 Developer friendly**

One of our goals in creating a time series platform is to optimize the value of the user's or developer's time. In other words, the faster they can solve the problem and get it up and running, the better the experience will be. This means that if we see users frequently writing code or creating projects to solve the same problem, we will try to introduce it into our platform or database. The less code developers have to write to solve their problems, the faster they can complete them.

**4.2 Time is special**

In addition to the obvious usability goals, we also saw that we could optimize the database around some features of time series. It's just insertion, we need to aggregate, we need to automatically exclude high-precision data in case the user wants to free up space. We can also build compression optimized for time series data. We also organize the data by indexing it for efficient queries. At the database level, we can get many optimizations.

**4.3 Going beyond the database makes development easier**

Another advantage of building specifically for time series is that we can go beyond databases. We found that most users encounter a common set of problems that need to be solved -- how to collect data, how to store data, how to process and monitor data, and how to visualize data.

**5. Problems solved by time series database compared with traditional database:**

Why do we use sequential databases when we already have traditional relational databases? It is usually because the performance of traditional relational database can not meet the needs of users. In addition, the time series data stored in the time series database has the default feature of sorting according to time order, which can be optimized in storage and query. In general, tens of millions of data will be generated in one second, and the aggregation calculation of tens of millions of data will be carried out. The following problems need to be solved in time series database: 1. Sequential data writing: how to support tens of millions of data points written per second. 2. Reading of sequential data: how to support the grouping and aggregation operation of hundreds of millions of data at the second level. 3. Cost sensitivity: The problem with massive data storage is cost. How to store these data at a lower cost will become the top priority to be solved in time series database.

Sequential database is different from traditional database in storage mode: traditional database adopts B-tree, which is beneficial to reduce the number of seek times during query and sequential insertion. We know that disk seek times are very slow, typically around 10ms. The random read and write of the disk is slower than the seek. Random write to B tree will consume a lot of time in the disk seek path, resulting in slow speed. The sequential database replaces B tree with LSM. LSM tree consists of data structures in memory and files on disk. Its working process is mainly: when data is written and updated, the data structure in memory is written first. WAL files are also written to avoid data loss. Data structures in memory are flushed to disk periodically or at a fixed size. The files on these disks will not be modified. As more files accumulate on the disk, the system periodically merges files to eliminate redundant data and reduce the number of files. The core idea of LSM Tree is to achieve higher write performance through memory write and subsequent sequential write to disk, avoiding random write.

**6. Conclusion:**

With the rapid development of Internet of Things technology, we will encounter more and more time series data in social life. Using a sequential database that specializes in processing this kind of data can greatly improve our data processing efficiency. It has great advantages in sequential data writing, query and data compression, and can solve many user pain points. However, the existing sequential database is still inadequate in storage, either stand-alone or difficult to maintain, and there is still a lot to be modified. It is very important to develop a more reasonable design of time series database and realize the time series database that can read and query more efficiently.