TSDB Features and Importance

A time series database (TSDB) is a software system optimized to sort and organize information measured by time. A time series is a collection of data points that are gathered at successive intervals and recorded in time order. Some examples of time series data (TSD) include changes to trades in a financial market, statistics collected from micro services, memory alerts, statuses, event data and dynamic assets.

Time series databases are especially useful to monitor access metrics, failure metrics, process behavior and workload monitoring. TSDB can sort through large and complex amounts of data, making the information more accessible than if it were stored in a traditional database.

1. **TSDB basic features**

Time-series business is vastly different from ordinary business in many ways, which are summarized as follows.

1. Continuous generation of massive amounts of data without peaks and valleys. For example, if the system monitors the indicators of 1w servers, each server collects 100 kinds of metrics per second, so there will be 100w TPS per second. This will also generate 300w TPS per second.
2. The data are inserted, and basically there is no update or delete operation. The data generated by the temporal business rarely has update and delete operations. Based on this fact, there will be great simplification in the design of temporal database architecture.
3. More recent data attention, the future will pay more attention to the flow of processing this link, time-sensitive data is rarely accessed, or even can be discarded. This is easy to understand, sentinel system we usually care most about the last hour of data, at most look at the last 3 days of data, rarely to look at the data before 3 days. With the arrival of streaming computing, time-series data in the future development will certainly be more concerned about the value of immediate data, and the value of this part of the data is undoubtedly the largest. It is a very common and important scenario that data can be alerted according to certain rules once it is generated, and the higher the timeliness of the alert, the better it is for the business.
4. There are multiple dimensional labels on the data, which often require multi-dimensional joint queries as well as statistical queries. Another very important function of time-series data is multi-dimensional aggregated statistical query, for example, the business needs to count the click rate and total revenue of advertisers google published in USA region in the last hour, which is a typical multi-dimensional aggregated statistical query requirements. This requirement is usually not high on the effectiveness requirements, but the query aggregation performance has relatively high requirements.
5. **TSDB core features**

the main technical points that TSDB needs to focus on are as follows.

1. High throughput write capacity. This is tailored to the characteristics of the continuous generation of large amounts of data for the timing business. To achieve a high throughput writing system, two basic technical requirements must be met: the system has horizontal scalability and a single LSM architecture. It is easy to understand that the system is horizontally scalable, and a single machine can not carry the load, the system must be clustered, and it must be easy to add nodes to expand. The LSM architecture is used to ensure high throughput for a single machine, where data is written to memory only and the write log is appended so that data no longer needs to be randomly written to disk, as is currently the case with HBase, Kudu, and Druid.
2. Data hierarchical storage/TTL, which is a technical feature tailored to the hot and cold nature of time-series data. Data hierarchical storage requires the ability to put the most recent hour-level data into memory, the most recent day-level data onto SSDs, and older data onto cheaper HDDs or just use TTL to expire out of existence.
3. High compression rate. Provide high compression rate has two aspects of consideration, on the one hand is the cost savings, which is easy to understand, will be 1T data compression to 100G can reduce 900G of hard disk overhead, which is a great temptation for the business. Another aspect is that the compressed data can be more easily guaranteed to be stored in memory, for example, the last 3 hours of data is 1T, I now only have 100G of memory, if not compressed, there will be 900G of data forced to put on the hard disk, then the query overhead will be very large, and the use of compression will be the 1T data into memory, query performance will be very good.
4. Multi-dimensional query capability. Time-series data usually has multiple dimensional labels to describe a piece of data, which is the dimensional column mentioned above. How to query efficiently according to several random dimensions is a problem that must be solved, this problem usually requires consideration of bitmap indexing or inverted indexing technology.
5. Efficient aggregation capability. A common requirement for timing business is to aggregate statistical report queries, for example, the Sentinel system needs to see the total number of exceptions on an interface in the last day, or the maximum time spent on an interface execution. Such aggregation is actually a simple count and max, the problem is how to efficiently query and aggregate the original data to meet the conditions based on such a large amount of data, knowing that the original value of the statistics may not be in memory because of the relatively long time ha, so this can be a very time-consuming operation. The industry's more mature solution is to use pre-aggregation, that is, to complete the basic aggregation operation when the data is written in.
6. Future technology points: anomaly detection in real time, future prediction, etc.
7. **Advantages of time-series databases over traditional relational databases**

Many people may think that adding a time stamp column to a traditional relational database will work as a temporal database. It is true that there is no problem when the data volume is small. However, temporal data is often generated by millions or even tens of millions of end devices, and the writing concurrency is relatively high, which is a massive data scenario.

MySQL has the following problems in the massive temporal data scenario:

1. High storage cost: poor compression for temporal data, which requires a large amount of machine resources.
2. High maintenance cost: single-machine system, which requires manual sub-banking and sub-tabling in the upper layer, and high maintenance cost.
3. Low write throughput: single machine write throughput is low, and it is difficult to meet the write pressure of ten million temporal data.
4. Poor query performance: suitable for transaction processing and poor performance for aggregation and analysis of massive data.

In addition, using Hadoop ecology (Hadoop, Spark, etc.) to store time-series data will have the following problems:

1. High data latency: offline batch systems, where data takes hours, or even days, from generation to analyzability.
2. Poor query performance: not making good use of indexes, relying on MapReduce tasks, and queries that take typically minutes.

Time-series database products are invented to solve the shortcomings and deficiencies of traditional relational databases in the storage and analysis of time-series data, and these products are uniformly categorized as time-series databases. Optimization of write, storage, query and other processes are carried out for the characteristics of time-series data, which are closely related to the characteristics of time-series data:

1. Storage cost: using the characteristics of time increment, dimensional repetition and smooth change of indicators, reasonable choice of coding compression algorithm to improve the data compression ratio; by pre-decreasing precision, do aggregation of historical data to save storage space.

(2) High concurrent write: batch write data to reduce network overhead; data is first written to memory, then periodically dumped to immutable file storage.

(3) Low query latency, high query concurrency: optimize common query patterns, reduce query latency through indexing and other techniques; improve query concurrency through caching, routing and other techniques.

**4. Why is a TSDB important**

Time series databases can help businesses monitor information in real time and address problems as they occur. They can also be used to predict future problems and prevent them before they happen.TSDB are more user-friendly and provide better write rates and stronger [query](https://searchsqlserver.techtarget.com/definition/query) performance despite the large amount of data they organize. In some ways, time series databases perform the same functionality as normal databases. However, trying to use a [relational](https://www.techtarget.com/searchdatamanagement/definition/relational-database) or [NoSQL database](https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL) for time series data would result in much slower and less efficient performance.

Current technology increasingly requires a need to query, stream and analyze information in real time. This includes a need for higher volumes, higher velocities and higher specificity in searching data. In recent years these demands have led to a strong and steady increase in the use of TSDB.

Querying in a time series database is similar to that in other kinds of databases, but instead of searching by values developers using a TSDB can search by a period of time that has passed, a date range or a particular point in time when an event happened.Some benefits of using a TSDB include:

1. The ability to scan extremely large quantities of data at once.
2. If data is collected every millisecond, the database can compress it to a minute or even shorter intervals.
3. TSDB use writable application program interfaces ([APIs](https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API)).