Overview of Elasticsearch and its relation to database

Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.[1]

Elastic allows you to store, search, and analyze large amounts of data quickly, in near real-time, and with multi-tenancy support. Elasticsearch is also developed in Java and uses Lucene as its core to implement all indexing and search features. It aims to make full-text search easy by hiding the complexity of Lucene through a simple RESTful API. However, Elasticsearch is more than just Lucene and full-text search. It is a distributed, real-time file storage where every field is indexed and searchable. It is a distributed, real-time analytics search engine. It is scalable to hundreds of servers, handling petabytes of structured or unstructured data. And all this functionality is integrated into a service that your application can interact with via a simple RESTful API, clients in various languages, or even the command line.[2]

However, with the rise of Elasticsearch, there is a question came up with now. Now almost all the information on the Internet says that the data is stored in the traditional database, and then a copy of the data is synchronized in Elasticsearch for retrieval, but none of it is very detailed about why it should be done and is it unnecessary to store two copies of the data when es itself can store the data? There are other issues that can arise. Although there is a fee and the supported syntax is not complete, now that Elasticsearch has support for SQL, I am getting confused about the boundary between Elasticsearch and the database. Elasticsearch does not support transactions but can ensure single data writes so that transactions can be implemented by code. It's hard to do a join query that can be implemented with wide tables like another NoSQL. Real-time performance can be tuned by configuration, while Elasticsearch is better in terms

of scalability and complex statistics. Based on the above questions, where is the difference or boundary between Elasticsearch and the database at this stage?[3] Actually, there are quite a lot of differences between Elasticsearch and traditional relation database:

They have different missions. A relational database is a database that uses a relational model to organize data, which stores data in the form of rows and columns to make it easier for users to understand [4]. Elasticsearch is a distributed open-source search and analytics engine for all types of data, including text, numeric, geospatial, structured and unstructured data. Overall, a relational database can store data and index it. A search engine can index data but also store it. [5]

They have different application scenarios. A relational database is more suitable for OLTP (a kind of computer application system with transaction element as the unit of data processing and human-computer interaction, the biggest advantage: the biggest advantage is that the input data can be processed instantly and answered in a timely manner) business scenarios; and Elasticsearch cannot be used as a pure database.

They have different storage types. The relational database generally only supports storing structured data. Characteristics of structured data are: Data is logically expressed and implemented by a two-dimensional table structure. Strictly follow the data format and length specification. Examples are bank transaction data, personal information data, etc. Elasticsearch supports both relational and unstructured data, e.g., JSON stored by the object or nested type or parent-child Join. Characteristics of unstructured data are irregular or incomplete data structure. Data that does not have a predefined data model and is not easily represented by a two-dimensional logical table in a database. Examples: including all formats of office documents, text, images, XML, HTML, various reports, images and audio/video information, etc.

They have different scalability. Relational database general problems, such as MySQL single table support data volume is limited, the volume of data will be divided into libraries and tables, and then considered distributed, the bottleneck of the native distributed as follows. The bottlenecks of native distribution are as follows, high business dependency, complex queries will be error prone. More importantly, distributed transactions cannot be handled effectively. This has given rise to many third-party NewSQL companies such as TIDB (open source and paid solutions). Elasticsearch supports horizontal scaling, inherently supports multi-node cluster deployment, has strong scalability, and even supports cross-cluster search; it can support PB+ data.

They solve different problems. Relational database for the core: add, delete, and change business scenarios, for full-text search will be slow to death; and Elasticsearch's inverted index mechanism is more suitable for full-text search. In actual business, if the data volume is not large, it is recommended to use a simple relational database combined with a simple SQL query to solve the problem. If you have no problem with performance, keep the architecture simple and use a single database for storage, with some caching (such as Redis) if necessary. If you are having performance problems with search, use a relational database in combination with Elasticsearch.

Overall, there are so many differences between traditional relational database and Elasticsearch in all kinds of aspects. There isn’t an one-shop solution. There is only the best suitable solution. We should choose the relational database or Elasticsearch based on the business requirement.

References:

[1] https://en.wikipedia.org/wiki/Elasticsearch

[2] https://www.elastic.co/what-is/elasticsearch

[3] https://elasticsearch.cn/question/8885

[4] https://www.oracle.com/database/what-is-a-relational-database/

[5] https://stackoverflow.com/questions/51639166/elasticsearch-vs-relational-database