> Bases de Datos 2, Resumen 1
> Miguel Ku Liang - 2019061913

Data in: documents and indices

Elasticsearch stores complex data structures that have been serialized as JSON documents. When a document is stored, it is indexed and fully searchable in near real-time. Elasticsearch uses a data structure called an inverted index that supports very fast full-text searches.

An index can be thought of as an optimized collection of documents and each document is a collection of fields. Elasticsearch indexes all data in every field and each indexed field has an optimized data structure.

Elasticsearch can be schema-less, meaning that documents can be indexed without explicitly specifying how to handle each of the fields that might occur in a document. When dynamic mapping is enabled, Elasticsearch detects and adds new fields to the index. Define rules to control dynamic mapping and explicitly define mappings to take full control of how fields are stored and indexed.

Information out: search and analyze

Elasticsearch provides a simple REST API for managing the cluster and indexing and searching the data.

Searching your data

The Elasticsearch REST APIs support structured queries, full text queries and complex queries that combine the two. Structured queries are similar to the queries in SQL. Full text queries find all documents that match the query string and return them sorted by relevance.

You can perform phrase searches, similarity searches and prefix searches, and get autocomplete suggestions.

Elasticsearch indexes non-textual data in optimized data structures that support high-performance geo and numerical queries.

Analyzing your data

Elasticsearch aggregations enable you to build complex summaries of the data and gain insight into metrics, patterns and trends. Aggregations lets you analyze and visualize data in real time. You can search documents, filter results and perform analytics at the same time, on the same data, in a single request.

Scalability and resilience

Elasticsearch is built to be always available and to scale with the needs. We can add servers to a cluster to increase capacity and Elasticsearch distributes the data and query load across all the available nodes. Elasticsearch knows how to balance multi-node clusters to provide scale and high availability.

An Elasticsearch index is just a logical grouping of one or more physical shards, where each shard is a self-contained index. There are two types of shards: primaries and replicas. Each document in an index belongs to

one primary shard. A replica shard is a copy of a primary shard. They provide redundant copies of the data to protect against hardware failure and increase capacity to serve read requests.

It depends...

The more shards, the more overhead there is. The larger the shard size, the longer it takes to move shards around when Elasticsearch needs to rebalance a cluster.

Querying lots of small shards makes the processing per shard faster, but more queries means more overhead, so querying a smaller number of larger shards might be faster.

Aim to keep the average shard size between a few GB and a few tens of GB and avoid the gazillion shards problem.

In case of disaster

A cluster's nodes need good connections to each other. To provide better connections, you co-locate the nodes in the same data center or nearby data centers. To maintain high availability, you need to avoid any single point of failure. Cross-cluster replication provides a way to synchronize indices from the primary cluster to a secondary remote cluster that can serve as a hot backup.

Cross-cluster replication is active-passive. The index on the primary cluster is the active leader index and handles all write requests. Indices replicated to secondary clusters are read-only followers.

Care and feeding

Security, monitoring and administrative features that are integrated into Elasticsearch enables you to use Kibana as a control center for managing a cluster.