Optimizing data retrieval by using MongoDb with Elasticsearch

Silvana Greca
Computer Science Dept.
silvana.greca@fshn.edu.al

Anxhela Kosta
Computer Science Dept
anxhela.kosta@fshn.edu.al

Suela Maxhelaku
Computer Science Dept
suela.maxhelaku@fshn.edu.al

Abstract

More and more organizations are facing with massive volumes of new, rapidly changing data types: structured, semi-structured, unstructured and polymorphic data. The performance for these organizations is directly dependent on the efficiency of data access, data processing, and data management. Data is storing in one environment that is called database. One database should be able to cover all the organizations database needs. Storing, indexing, filtering and returning data on real-time, are four very important issues that affect directly at high performance. But, how to get these features in one database? You can optimize data by using MongoDb combined with Elasticsearch. MongoDb is a NOSQL database for storing big data and returning data on real-time. ElasticSearch is the best for indexing and filtering data. The purpose of this paper is to demonstrate combination of using MongoDb with Elasticsearch for optimize data and how to evaluate their performance according to the typical use for storing and retrieving data. This demonstration is focus on creating a "Data warehouse" for Agricultural products in Albania.

Keywords:

Elasticsearch, MongoDb**,** Performance, Optimizing data, Agriculture

1. Introduction

The amount of data that businesses can collect is really enormous and hence the volume of the data becomes a critical factor in analyzing them. Business use different database to store data, but it can be difficult to choose the right DBMS, or NOSQ, so they based on requirements related to the quality attributes Consistency, Scalability and Performance [Nilsson17].

Today on modern applications, most of developer use NOSQL technologies to provide superior performance because NOSQL databases are built to allow the insertion of data without a predefined schema. There are many different NOSQL systems which differ from each other and are develop for different purposes. In Albania, the Ministry of Agriculture, Rural Development has the necessary for tracking and analyzing agricultural products data, but these data are in different source systems [Instat17].

It is necessary to use some tools and methods to integrate this data. Agricultural products in Albania need a Data Warehouse. Data Warehouse is typically the kind of database that integrates, collects and analyzes data found on different source systems. Since the data types are structured, semi-structured, unstructured for agricultural we created a use case on MongoDb as a "Data warehouse".

For Ministry of Agriculture, Rural Development is very important to have a system that indexing and filtering data at high performance and return the statistics for import and export of agricultural products in Albania. For these we have combined MongoDb with Elasticsearch a full-text search engine written in Java, that analyze large amounts of data on the fly.
Combining MongoDb with Elasticsearch, the Ministry of Agriculture in Albania will have a data warehouse for store a large amount of unstructured data and high performance on searching query and filtering data.

2. Agriculture challenge

Storing and processing data for agriculture products in Albania is a central task. Agricultural products are divided on four groups:

- arboriculture
- plants and fields
- farming
- fishing

For each group the Ministry of Agriculture, Rural Development has the necessary to has, statistics on import, export, agriculture products (monthly, year, etc.), the international price of products (daily, weekly) information, on the production of agricultural products etc. For example, is necessary to know how many exports were made in 2010 from Albania's supplier to Italy in millions of Lek.[1]

Data of agriculture products are storing on distributed databases, so it has low performance to have the statistics on real time.

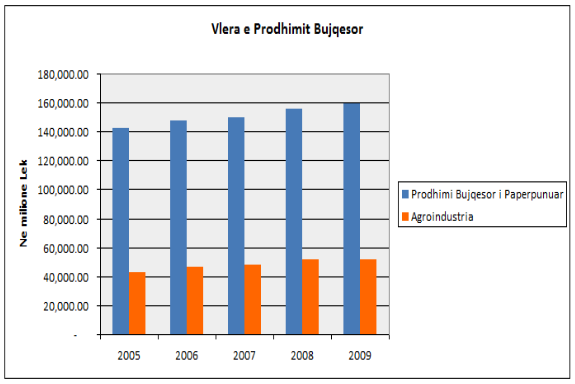


Figure 1: The value of agricultural products

2.1 Use case

The Ministry of Agriculture Rural Development to get data on distributed database with a high performance needs to have a Data Warehouse for collected all data in one environment. Data warehousing is a technology that aggregates structured data from one or more sources so that it can be compared and analyzed for greater business intelligence. Data warehouses are designed to give a long-range view of data over time. They trade off transaction volume and instead specialize in data aggregation.

The Ministry of Agriculture Rural Development requests information to produce statistics on exports that have been made over the years and which countries are cooperating, how many products are in millions of lek[1]?

First, we have identified and have collect requirements, design the dimensional model and execute T-SQL queries to create and populate the dimension and fact tables. We have created the Export (faktEksport) fact table that constructed with respective dimension tables: *Products, Time, supplier and Clients.*

```
Create table faktEksport
(produktId int not null references DimProdukt (produktId),
KlientiId int not null references DimKlientet (KlientId),
FurnitorId int not null references DimFurnitoret (FurnitorId),
KohaId int not null references DimKoha (KohaId),
SasiaProdukteve int not null,
VleraMonetare money not null,
Constraint [Pk_faktEksport] primary key nonclusterd
( [produktId], [KlintiId], [FurnitorId], [KohaId]
)
)
```

Figure 2. T-SQL Script for fact table

Now, to design the Relational Databases we have used Star schema, due to hierarchical attribute model it provides for analysis and speedy performance in querying the data.

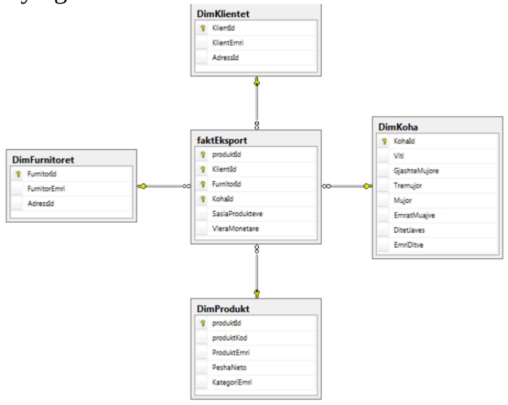


Figure 3. The star schema

[1] Lek- the international currency of Albania

2.2 Advantage and disadvantage of Data warehouse

The implementation of a data warehouse for the Rural Sector in Albania can bring major, benefits for Ministry of Agriculture Rural Development including:

- provides systematic and periodic information for all agriculture sectors,
- the government increases profits in the market share and lowers the cost of providing information,
- increases scientific research performance and effectiveness to benefit farmers as well.

The other benefits of a data warehouse are the ability to analyzing data from multiple sources. Data warehouses are very important tool for big data.

Since agriculture data are unstructured, complex and unprocessed, data warehouse finds it difficult to excel with high performance. Also, if you want to scale data warehouse it is very expensive.

This is a big disadvantage, because it is very important to find the best idea for optimizing data and get data on real-time for agriculture sector. Data warehouse is built on Sql Server, a relation database system (RDBMS) that allow the insertion of data only on predefined schema. So, is a big complex to manage the unstructured data.

3 The solution- Nosql

Today, the number of different databases is increases rapidly, but you have to choose the one that cover all your organizations database needs.

NoSQL databases are unstructured, they do not have a fixed schema and their usage interface is simple, allowing developers to start using them quickly. NoSQL databases are often highly optimized for retrieve and append operations. They are the concept for storing and managing data that has problem concerning unstructured data, and horizontal scalability. Nosql implements features that traditional database management system, do not.

As we have mentioned at previous paper, there are many different NoSQL systems which differ from each other and are developed for different purposes [Greca18]. According to the classification Nosql are divided on four categories [Corbellin16]:

1. Key-value- These databases allow storing data under a key. They work similarly to a conventional hash table, but by distributing keys (and values) among a set of physical nodes.
2. Wide Column or Column Families- Instead of saving data by row (as in relational databases), this type of databases store data by column. Thus, some rows may not contain part of the columns, offering flexibility in data definition and allowing to apply data compression algorithms per column. Furthermore, columns that are not often queried together can be distributed across different nodes.
3. Document-oriented: A document is a series of fields with attributes, for example: name= “Claus”, lastname=”Costa” is a document with 2 fields. Most databases of this type store documents in semi structured formats such as XML (eXtensible Markup Language), [Shanmugasundaram99], JSON (JavaScript Object Notation), [Nurseitov09]. They work similarly to Key-Value databases, but in this case, the key is always a document's ID and the value is a document with a predefined, known type (JSON or XML) that allows queries on the document's fields.
4. Graph-oriented: These databases aim to store data in a graph-like structure. It is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data.

NoSQL databases are those databases that are non-relational, open source, distributed in nature as well as it is having high performance in a linear way that is horizontally scalable. Non-relational database does not organize its data in related tables. NoSQL databases are open source; therefore, everyone can look into its code freely, update it according to his needs and compile it. Distributed means data is spread to different machines (sources) and is managed by different machines so here it uses the concept of data replication Nosql, are very popular today.

There are a huge number of Nosql database, but for our use case we are using MongoDb [2]. Our choice of MongoDb is motivated by the need for a document-oriented store for the statistic of import export data for agriculture product.

3.1. **MongoDB**

MongoDb [MongoDb18] is a database between relational databases and non-relational database, its features are:

1) it is non-relational database, which features the richest and most like the relational database;

2) support complex data types: MongoDb support bjson data structures to store complex data types;

3) powerful query language: it allows most of functions like query in single-table of relational databases, and also support index.

4) High-speed access to mass data: when the data exceeds 50GB, MongoDb access speed is 10 times than MySQL. Because of these characteristics of MongoDb, many projects with increasing data are considering using MongoDb instead of relational database [Greca17].

Due to the features and strengths of MongoDb has been adopted as backend software by a number of major organizations including Craigslist, eBay, Foursquare, Source Forge, Viacom, and the New York Times among others MongoDb is the most popular NoSQL database system. As mentioned earlier, one of the important features of MongoDb that has been utilized in this paper is the ability to store unstructured big data documents and returned them on real-time.

We can say that MongoDb is the best solution to store agriculture product. MongoDb use find() query for looking to the document, on real-time. Even mongoDb is superior for large, flexible data storage, we can show that its search capabilities aren't as robust as we expect.

For this reason, we have identifying important research directions, literature review focusing on the main goal to archive best way on keyword searching unstructured and distributed data.

According to [Nilson17], we have to use an open source search engine called Elasticsearch which is suitable for performing full-text searches on a wide variety of different information. Elasticsearch is part of a software stack called the Elastic Stack which also offers extra functionality such as visualization, security, and performance monitoring.

3.2 Elasticsearch

Elasticsearch is an open-source, RESTful, distributed search and analytics engine built on Apache Lucene. Elasticsearch is release in 2010, and has quickly become the most popular search engine, and is commonly used for log analytics, full-text search, security intelligence, business analytics, and operational intelligence use cases [Elasticsearch18].

With elasticsearch we can perform and combine many types of searches- structured, unstructured, geo, metric data. The JSON documents we can send to Elasticsearch using the API. Elasticsearch automatically stores the original document and adds a searchable reference to the document in the cluster's index.

To understand how the Elasticsearch query works see the figure below:

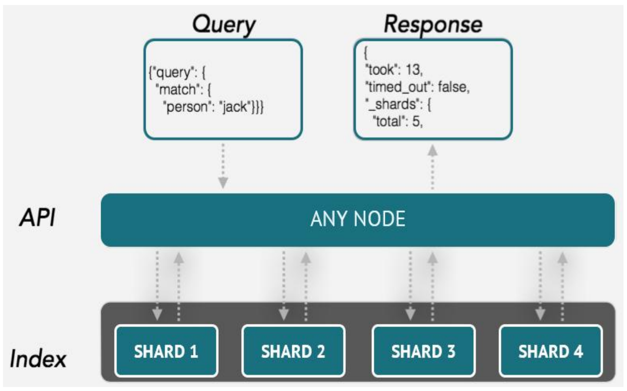


Figure 4. Elasticsearch query works

Below, a short description of the most important elements we have to know for elasticsearch **[**Elasticsearch18].

Index- is a collection of documents that have similar characteristics. An index is identified by a unique name that refers to the index when performing indexing search, update, and delete operations

Node- is a single server that holds some data and participates on the cluster's indexing and querying. A node can be configured to join a specific cluster by the particular cluster name.

Cluster- is a collection of one or more servers that together hold entire data and give federated indexing and search capabilities across all servers

Shard- is a subset of documents of an index. An index can be divided into many shards.

3.2.1 Benefits of using Elasticsearch

According to **[**Kononenko14], elasticsearch has some advantages:

- Manages the huge amount of data: As a comparison to the traditional SQL database management systems that take more than 10 seconds to fetch required search query data, Elasticsearch can do that within 10 ms.
- Direct, Easy and Fast access: Documents are stored in a close proximity to the corresponding metadata in the index. This reduces the no of data reads and as a result increases the search result response.
- Scalability of the search Engine: As Elasticsearch has a distributed architecture it enables to scale up to thousands of servers and accommodate petabytes of data. The customers then need not manage the complexity of distributed design as it has been done automatically.

These benefits make Elasticsearch a natural fit with MongoDb, the most popular Big Data system.

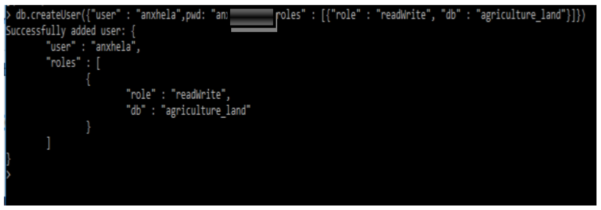
4. Mongodb, Elasticsearch, working together

MongoDb is a great general-purpose database but one place where its limitations show up is with its full text searching [IBM18]. The mongoDb challenge will be solved by combining it with Elasticsearch. They work well together. Elasticsearch is the best engine, on filtering and searching text.

In order to demonstrate the practical use of this combination, we have created a database for agriculture data on mongoDb, then mapping with elasticsearch using a compose transporter. We have used MongoDb for storing agriculture data and Elasticsearch for searching the data.

4.1 The Demonstration

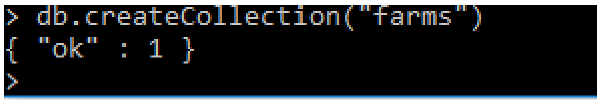
At MongoDb, first we create the database called agriculture_land. When connecting to multiple databases, we can add the user by logging into MongoDb as an admin user and running the following command**.**



```
> db.createUser({"user" : "anxhela",pwd: "an[redacted]roles" : [{"role" : "readWrite", "db" : "agriculture_land"}]})
Successfully added user: {
        "user" : "anxhela",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "agriculture_land"
                }
        ]
}
>
```

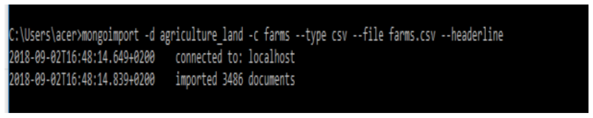
Figure 5. Create user

Create one collection Farms and populate it with data by importing a csv file.



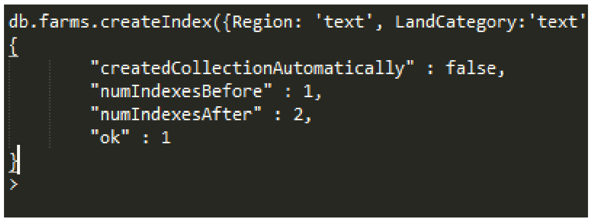
```
> db.createCollection("farms")
{ "ok" : 1 }
>
```

The csv file has 3486 documents, with attributes (State, LandCategory, Region or State, Year, Acre Value). Due to the confidentiality for the Ministry of Agriculture, Rural Development, data using on this demonstration are test.



```
C:\Users\acer>mongoimport -d agriculture_land -c farms --type csv --file farms.csv --headerline
2018-09-02T16:48:14.649+0200    connected to: localhost
2018-09-02T16:48:14.839+0200    imported 3486 documents
```

Create a text index in both Region and LandCategory fields of the farms collection.

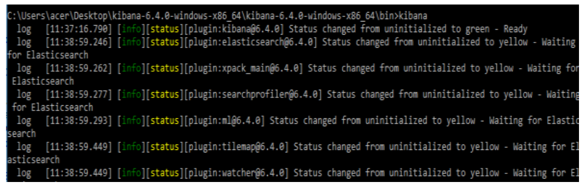


```
db.farms.createIndex({Region: 'text', LandCategory:'text'
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>
```

Now we need to find, which is the ace value for one region, one year-by searching. The first part of the problem to solve on MongoDb, is how to efficiently search for items in the agriculture product. So, we index mongoDb at Elasticsearch.

The Elasticsearch server is easy to install, and the default configuration supplied with the server is sufficient for a standalone use without tweaking, but if you want you can tune some of the parameters.

We have install Kibana [Beaver15], an open-source visualization tool, with Elasticsearch to visualize data and build interactive dashboards.



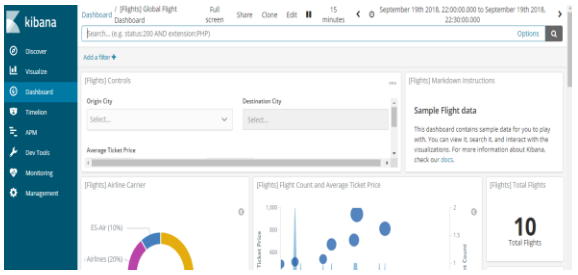
```
C:\Users\acer\Desktop\kibana-6.4.0-windows-x86_64\kibana-6.4.0-windows-x86_64\bin>kibana
  log   [11:37:16.790] [info][status][plugin:kibana@6.4.0] Status changed from uninitialized to green - Ready
  log   [11:38:59.246] [info][status][plugin:elasticsearch@6.4.0] Status changed from uninitialized to yellow - Waiting
for Elasticsearch
  log   [11:38:59.262] [info][status][plugin:xpack_main@6.4.0] Status changed from uninitialized to yellow - Waiting for
 Elasticsearch
  log   [11:38:59.277] [info][status][plugin:searchprofiler@6.4.0] Status changed from uninitialized to yellow - Waiting
 for Elasticsearch
  log   [11:38:59.293] [info][status][plugin:ml@6.4.0] Status changed from uninitialized to yellow - Waiting for Elastic
search
  log   [11:38:59.449] [info][status][plugin:tilemap@6.4.0] Status changed from uninitialized to yellow - Waiting for El
asticsearch
  log   [11:38:59.449] [info][status][plugin:watcher@6.4.0] Status changed from uninitialized to yellow - Waiting for El
```

Figure **6**. Configure Elasticsearch using Kibana

After configure elasticsearch we use compose transporter [IBM18]- open sources that lets to getting start up and connect Elasticsearch with mongoDb. A combined system using MongoDb as primary data store and Elasticsearch as secondary data store, could be used to achieve fast full-text search queries for all types of expressions, simple and complex.

By using this combination, we can: completes complex search queries quickly, find data fast by using indexes, and return all data on real- time with high performance.

The figure below shows a visual representation of Kibana, where data of agriculture products can be filtering for all attributes and then can be retrieving with high performance.



Sample Flight data

This dashboard contains sample data for you to play with. You can view it, search it, and interact with the visualizations. For more information about Kibana, check our docs.

Figure 7. Kibana elasticsearch monitoring

Conclusion

Today, MongoDb databases are becoming more popular because the need for easily scalable, dynamic databases is growing, when you are dealing with cubes and huge volume of data. The full-text search queries need to be fast and therefore executed in a matter of seconds or even milliseconds. Elasticsearch is an easily scalable, full-text search engine that is capable of handling large amounts of online and schema-less data. We conclude that combined Elasticsearch with MongoDb is the perfect choose for organizations to store big, complex data and searching text on it. The demonstration established, indeed facilitated the whole process of literature review gathered in this regard. The future for elasticsearch and mongoDb is consistent and can help achieve faster performance with search results for organizations search engine.

References

[Nilson17] Ch. Nilsson. J. Bengtson. *Storage and Transformation for Data Analysis Using NoSQL*, 80: 3–25, 2017

[Greca18] S. Greca, A.Kosta, S.Maxhelaku. NRA-CSIT- *Application on Nosql database: Classification, Characteristics and Comparison*, 2-6, 2017

[Shanmugasundaram99] J. Shanmugasundaram, K.Tufe, G.HE. *Relational Databases for Querying XML Documents: Limitations and Opportunities*, 13, 1999

[Nurseitov09] N. Nurseitov, M.Paulson, R.Reynolds, C.Izurieta. *Comparison of JSON and XML Data Interchange Formats: A Case Study*, 3-6, 2019

[Beaver15] D. Beaver, S. Hutchison. *Elasticsearch, Logstash, and Kibana (ELK*), 35, 2015

[Corbellin16] A.Corbellini, C.Mateos, A. Zunino, D.Gody, S.Schiaffino. *Persisting big-data: The NoSQL landscape*, 23: 4-10, 2016

[Instat17] Instat. *Vjetari statistikor rajonal*, 131: 103-129, 2017

[MongoDb18] Mongodb Center. *What is MongoDb*: https://www.mongodb.com/what-is-mongodb

[Elasticsearch18] Elasticsearch, 2018:

https://www.elastic.co/products/elasticsearch.

[IBM18] Compose, an IBM Company Mongoosastic. *The Power of MongoDB & Elasticsearch Together, 2018*