Comparing Alternative Storage Models for Words Extracted from Legal Texts

Ana Paula Sodré[1], Luis Eduardo Mochenski Floriano[1],Dimmy Magalhães[1]
Cristina D. Aguiar[2], Aurora Pozo[1], Carmem S. Hara[1]

[1]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR

[2]Universidade de São Paulo, São Carlos, Brazil

{apas19,lemf19,dksmagalhaes,aurora,carmem}@inf.ufpr.br, cdac@icmc.usp.br

Abstract. *The COVID-19 pandemic created new demands for services in the judicial system, requiring the use of a data warehouse (DW). Although there exist approaches that use DW in the judicial domain, few target the pandemic or publicly provide the information extracted from the texts. Following the needs of a legal expert, we have developed the COVID-19 Portal. It extracts documents from the Supreme Federal Court in Brazil to obtain quantitative information on words used in the texts. In this paper, we present the design of a DW, and show the query performance improvement achieved with its implementation. The DW has been developed on Postgres, and its performance is compared with the original implementation on MongoDB Cloud and a local MongoDB database.*

1. Introduction

The COVID-19 Pandemic has deeply affected all sectors of society. This also occurs in the judicial area, in which the number of legal proceedings involving the disease has grown exponentially. To provide a qualitative analysis of texts in decisions and lawsuits related to the pandemic, and addressed by the Federal Supreme Court (STF) in Brazil, we are developing the COVID-19 Portal[1]. The portal collects judicial processes (and their associated documents) from the STF Portal[2] that contain the terms *pandemia* or *covid*. It applies natural language processing (NLP) and artificial intelligence techniques to make quantitative and semantic analysis on the collected data.

All Portal data is stored on MongoDB Cloud[3] in its free version. MongoDB is a NoSQL database, and the Portal uses JavaScript Object Notation (JSON) format for storage. Given the fact that the NLP output is already created as JSON objects, using a NoSQL database seemed to be a good strategy to avoid complex post-processing before persisting the data. Furthermore, the structure of the objects have been defined based on the specific queries required to feed the charts and graphs of the Portal, aiming at providing a low response time. However, this approach makes it almost impossible to extend the Portal with new functionalities.

This motivated us to investigate alternative storage models, i.e., to investigate how to organize and store data from the documents focusing on the analysis required

[1]http://portalcovid-cbio-cd.herokuapp.com/
[2]http://portal.stf.jus.br/
[3]https://cloud.mongodb.com/

by the specialists and providing efficiency. Data warehousing is an appropriate technology to be used for this purpose [Chaudhuri and Dayal 1997]. Its core component, the data warehouse (DW), stores subject-oriented, integrated, historical, and non-volatile data [Kimball and Ross 2013]. The historical characteristic is relevant in judicial processes since it guarantees the investigation of the progress of the processes over time. Furthermore, the subject-oriented characteristic provides support to focus on the subjects of interest of specialists like lawyers and judges. Regarding efficiency, the performance of the queries is very important to guarantee an appropriate use of the Portal.

In relational implementations of the DW, data is organized in a star schema, where a central fact table is linked to several satellite dimension tables, thus resembling a star. The fact table stores numeric measures representing the subjects of interest (i.e., facts) and references to the dimensions. A dimension table stores alphanumeric attributes that describe its characteristics. Furthermore, fact tables that share common dimension tables may be organized in a more complex structure called fact constellation. Analytical queries issued against the fact constellation require the processing of the star join operation, i.e., performing joins between each fact table and each dimension table involved in the queries [Rocha and Ciferri 2020].

In this paper, we describe the DW that we developed to support the COVID-19 Portal. It is organized as a fact constellation and implemented using the PostgreSQL[4] relational database management system. Motivated by the requirements defined by the specialists, its objective is to provide support for performing analysis associated with the frequencies and proximity of the words that appear in the documents. We also provide a comparison considering the two proposed approaches for querying data: the new approach proposed in this paper and based on the use of a DW and the existing approach based on the use of the NoSQL database.

This work is organized as follows. Section 2 reviews related work and Section 3 describes the COVID-19 Portal. The design and implementatin of the DW is the subject of Section 4. Section 5 describes the experimental results, and Section 6 concludes the paper.

2. Related Work

Several studies present the main differences between NoSQL and relational databases using cost, volume and scalability as comparison criteria. Despite the growing popularity of NoSQL, there are many aspects to be considered when choosing a database model.

Kunda and Phiri (2017) highlight how easy it is to implement robust, consistent and secure applications on relational databases. However, the relational model relies on a fixed schema. The study also shows that NoSQL databases perform well when consider large amounts of data, but it is less consistent and more vulnerable. Furthermore, as mentioned in [Mohamed A. Mohamed and Ismail 2014], NoSQL databases are not designed for DW applications given that designers have focused on aspects such as high performance, scalability, and availability. However, their ability to store big volumes of data may benefit DW applications by supporting datasets with increasing size.

In the area of law, there are case studies in which a DW is built using legal data. In

[4]https://www.postgresql.org/

[Bruzarosco et al. 2000], a data mart prototype with dimensional modeling is created to assist in the administration of a legal portal. In their system, client queries are forwarded to the associated professional who is most qualified on the subject. The DW stores in the fact table these queries, while customer, technical area, and date are some of the dimension tables. Unlike their work, the COVID-19 Portal considers data related to processes within the pandemic context. There are two fact tables, which store the frequency and the proximity of words, while some of the dimensional tables are the location, the legal process, and the minister handling the process.

From a functional standpoint, Datalawyer Insights[5] is the work that comes closest to ours. As our Portal, it is focused on lawsuits related to the pandemic in Brazil. However, one of the differences found is the source of the data. While the Datalawyer uses data from the Labor Justice Court, we consider processes provided by the Federal Supreme Court. Furthermore, they focus on the quantitative approach to data analysis. On the other hand, in addition to quantitative analysis, our Portal also processes documents at a more refined level to calculate the proximity of words and categorize them.

3. The COVID-19 Portal

Most of the analysis performed on the COVID-19 Portal are based on the words present in the texts: their frequency and their proximity in the texts. Words that are frequently used together may help specialists develop arguments for new processes related to the pandemic. Furthermore, the Portal presents a general process clustering and a clustering based on STF ministers. The first helps the user in understanding how the court documents are related to each other, and the second may show the legal guidelines of each STF minister. For example, it can highlight possible unanimous votes given a particular theme. All the information in the Portal may be shown regarding the entire country, or a specific state, or a particular field of law.

The data stored on the MongoDB NoSQL database compose two main collections:

- *ProcessoSTF*: holds information about each process, such as location, ministers, and classes;
- *pdflinks*: contains links for all documents of a certain process, as well as the most common words in their texts.

Both collections can have multiple documents whose fields are composed of different types, such as texts, numbers, lists, dictionaries and also undefined values. In ProcessoSTF there are sixty one different fields while in pdflinks there are eighteen. It is important to mention that in our implementation, there was no use of a schema, so the number of fields can vary among the documents of a collection.

To query the collection, there are no declarative languages such as SQL. In fact, MongoDB defines a query as a pipeline of stages, where each stage defines a transformation on the data. In comparison to SQL, it is difficult to maintain the pipeline, given that modifications of a middle state may affect the following ones. On the other hand, each stage may be updated in real-time, which allows queries to be built incrementally.

[5]https://www.datalawyer.com.br/dados-covid-19-justica-trabalhista

4. The Portal Database Project

Most of the information in the Portal is based on the analysis of word frequency and proximity, which are shown based on some user input parameters such as the state of interest or a particular legal class. It motivated us to model the DW with a fact table to represent the results of the analysis, and dimension tables for the current and other possible filter options that may be implemented in the Portal.

The proposed schema of the fact constellation is depicted in Figure 1. Fact tables are represented using thick lines and yellow. Each fact table is described in terms of the numeric measure that it represents and a primary key to each linked dimension table. The fact table's primary key is a combination of the primary keys of the dimension tables. Dimension tables are represented using thin lines and blue. Each dimension table contains a primary key and several descriptive attributes.

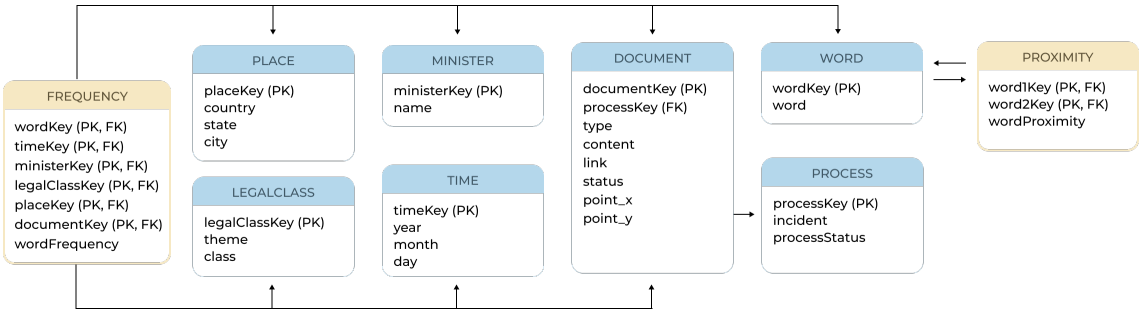

Figura 1. Fact constellation of the data warehouse of the COVID-19 Portal.

The schema contains the following fact tables:

- *FREQUENCY*: holds the frequency of a given word in a given time and place, considering a given minister, legal class, and document;
- *PROXIMITY*: contains the proximity factor of two words in the dimension table *WORD*.

There are seven dimension tables:

- *TIME*: stores the time at which the process has been filed;
- *PLACE*: contains the place where the process has been file;.
- *WORD*: holds the most common words present in each text of each process;
- *PROCESS*: store data related to legal processes;
- *DOCUMENT*: contains data of the documents. Several documents can compose a legal process, and words are counted in each document separately;
- *LEGALCLASS*: contains the classification of each process, which can be categorized by a *class* and a *theme*;
- *MINISTER*: stores the ministers of the STF responsible for the process.

We have implemented the schema depicted in Figure 1 in a relational database. Although the NLP tools output data in JSON and storing the results in a relational database requires a more elaborated post-processing, it is much easier to create a generalized SQL query for consulting than it is with a NoSQL approach [Kunda and Phiri 2017].

In Figure 2, we depict the skeleton of a query that sums up the frequency of a word, generalizing the possible conditions (i.e., filters) related to the dimensions. Because of the

```
SELECT SUM(wordFrequency)
FROM (SELECT DISTINCT F.documentKey, F.wordFrequency
      FROM FREQUENCY AS F NATURAL JOIN <DIM>
      WHERE <CONDITION>)
```

Figura 2. Fact table generic query

star join operation, each join in this query represents the connection of the fact table and a dimension related to it. According to the proposed fact constellation, it is possible to investigate the frequency of a word considering several perspectives. For instance, it is possible to investigate frequency by word and time or frequency by document and place.

5. Experimental Results

In this section we report an initial experimental analysis comparing the original implementation of the Portal with MongoDB with our DW implementation. We have adopted PostgreSQL and used ORM Sequelize[6] to handle the creation of queries and delegate the queries to PostgresSQL. Sequelize is based on node.js[7] to provide data to the Portal Web page. Regarding MongoDB, we used mongoose[8] to handle NoSQL queries and submit them to it. We also consider two instances: the original MongoDB Cloud and a local installed MongoDB database. The size of the DW is 164.31MB, with 446,700 tuples in the *FREQUENCY* fact table, and 473,710 tuples in the *PROXIMITY* fact table. The size of the the MongoDB database is 156MB.

```
SELECT SUM(wordFrequency)
FROM (SELECT DISTINCT F.documentKey, F.wordFrequency
      FROM FREQUENCY AS F
           NATURAL JOIN MINISTER M
           NATURAL JOIN WORD W
           NATURAL JOIN PLACE P
      WHERE M.name = <MINISTER NAME> and
           W.word = <WORD> and
           P.state = <STATE>)
```

Figura 3. Query created for the experimental study

We created a query similar to the one used in the Portal, as shown in Figure 3. It returns the number of times a word is used in the processes of a given minister and in a given state. In this scope we have instantiated Query 1 and 2, as follows.

- *Query 1*: considers the word ‘pandemia’ for Marco Aurélio as minister, located at São Paulo.
- *Query 2*: considers the word ‘covid-19’ for Edson Fachin as minister, located at Federal District.

[6]https://sequelize.org
[7]https://nodejs.org
[8]https://mongoosejs.com

Although both queries are similar, creating the same ones in NoSQL is not as trivial. As mentioned in [Kunda and Phiri 2017], NoSQL databases are usually based on an object-oriented API for data manipulation and each implementation of this type of database has its own data manipulation language. In this sense, it was harder to create the query in NoSQL during the development of the tests, precisely due to the adaptation to the language syntax. For the runtime analysis, we ignored the first 5 connections of each query. We reported the average execution time of the 5 subsequent connections.

Table 1 shows the time spent to process Query 1 and Query 2 considering our implementation (PostgreSQL column), the original MongoDB Cloud, and the local installed MongoDB database. The performance results show that queries issued in PostgreSQL greatly overcame the queries carried out in MongoDB. Query 1 executed in PostgreSQL was 12 times faster than the local MongoDB. This difference was even greater considering the online MongoDB: PostgreSQL was 51 times faster. Considering Query 2 and its execution in PostgreSQL, it was 3 times faster than the local MongoDB and 57 times faster than the online MongoDB. It is also important to note that when executed online, which is the approach the Portal currently adopts, MongoDB was up to 4 times and 18 times slower than its local version for Query 1 and Query 2, respectively. Finally, the difference in the time spent between Query 1 and Query 2 is related to the difference in the number of occurrences of the searched word. The word 'pandemic' (Query 1) appeared 50 times in the documents, while 'covid-19'(Query 2) appeared only 29 times.

	PostgreSQL	**MongoDB (local)**	**MongoDB (online)**
Query 1	64,74 ms	789 ms	3,33 s
Query 2	44,08 ms	139,6 ms	2,55 s

Tabela 1. Performance of the queries in PostgreSQL and MongoDB

We can conclude that, for the queries currently available in the Covid-19 Portal, the proposed DW implementation guarantees faster data retrieval to the Web application. Therefore, specialists can obtain charts and graphs more quickly, helping them focus on the subject at hand and enhancing their experience. Furthermore, the use of generalized queries helps developers to create new features for the front-end project, such as filters, as well as to maintain the integration with other relational data sources.

6. Conclusion

This paper reports on an alternative DW storage for the COVID-19 Portal, which currently adopts MongoDB Cloud as its storage platform. We present the design of the DW. The proposed fact constellation goes a step forward because it considers as perspective of analysis data related to word, time, place, legal class, minister, and document. This flexibility will allow us to extend the filter options currently available in the Portal, providing more functionalities for the specialists. We also compare the relational implementation of the DW using PostgreSQL with the MongoDB implementation. The first noticeable factor between the NoSQL structure and a relational DW approach is the query manipulation. As shown in previous studies, creating a generalized query to a non-relational base is way more complicated than in the relational base. Furthermore, the tests showed a significant difference in query performance. Future work also includes the execution of different types of query.

Referências

Bruzarosco, D. C., Castoldi, A. V., and Pacheco, R. C. d. S. (2000). Developing data warehouse using dimensional model. *Acta Scientiarum*, 22:1389–1397.

Chaudhuri, S. and Dayal, U. (1997). An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74.

Kimball, R. and Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley, 3 edition.

Kunda, D. and Phiri, H. (2017). A comparative study of NoSQL and relational database. *Zambia Information Communication Technology (ICT) Journal*, 1.

Mohamed A. Mohamed, O. G. A. and Ismail, M. O. (2014). Relational vs. NoSQL databases: A survey. *International Journal of Computer and Information Technology*, 3.

Rocha, G. M. and Ciferri, C. D. A. (2020). Efficient processing of analytical queries extended with similarity search predicates over images in spark. *Journal of Information and Data Management*, 11(3):209–227.