EasyChair Preprint

№ 10735

Data Mapping and Querying in NoSQL Data Warehouses

Malika Taouai and Faiza Ghozzi

August 18, 2023

Data Mapping and Querying in NoSQL Data Warehouses

Malika Taouai[1,2] and Faiza Ghozzi[1,3][0000−0002−6970−0297]

[1] Sfax University, Higher Institute of Computer Science and Multimedia, BP 1030, Tunisia
[2] taouai.malika@isimsf.u-sfax.tn
[3] MIRACL Laboratory faiza.ghozzi@isims.usf.tn

Abstract. In the era of big data, traditional databases struggle to handle the volume variety and velocity of data. NoSQL systems present a promising alternative providing faster data access enhanced scalability, and increased flexibility. This research paper introduces a practical approach that involves a set of mapping rules to adapt conceptual multidimensional schemes to NoSQL document-oriented representations enabling efficient online analytical processing (OLAP) analytics. Our approach focuses on two models the embedded document model which allows for storing data directly within the document for easy context-based viewing and the reference document model which offers flexibility by storing data separately and utilizing references as needed. Consequently, this work presents two notable contributions, one for each model employing embedded document and reference documents. To evaluate the efficiency and effectiveness of both models. A performance comparison was conducted employing query execution time as a key metric for OLAP analytics. The insights gained from this evaluation shed light on the performance and suitability of each model in different kind of queries.

Keywords: Big Data · Data Warehouse · Multidimensional Database · NoSQL · Document-oriented · Snowflake schema

1 INTRODUCTION

With the growth of modern businesses, organizations have access to an unprecedented volume of data from various sources. While the abundance of information is advantageous, effectively organizing, analyzing, and making data-driven decisions pose significant challenges.

The normalized and structured data that a relational database (RDB) stores in tables, is later considered a limitation due to the rapid growth of applications. RDB cannot manage enormous amounts of denormalized data, which is why companies like Google, Facebook, and Amazon have chosen to store their data in NoSQL databases [1]. A solution that supports the subject-oriented model better than RDB can be found in NoSQL databases [2].

With the increasing adoption of NoSQL databases, it becomes crucial to find a suitable approach for mapping conceptual multidimensional structures to NoSQL structures. This conversion is essential to enable data integration, a common requirement in contemporary applications.

To address these changing requirements, the document-oriented nature of NoSQL databases aligns well with the needs of multidimensional data warehousing. Furthermore, optimizing query performance and identifying the most suitable logical schema for efficient analysis are key considerations for researchers in this field.

A set of rules to facilitate the mapping of snowflake schemas and their optimization structure to a logical NoSQL document-oriented representation. We present two contributions, one for each model, employing embedded-document and reference documents. We compare the performance results of both models during execution time to evaluate their effectiveness.

This paper is organized as follows. Section 2 represents a related work. Section 3 gives a formal representation of the multidimensional schema. Section 4 presents a formal representation of the NoSQL Document-oriented Databases. Section 5 details the transformation rules. Section 6 represents experiments and discussion. Section 7 concludes the paper and draws future research directions.

2 Related Work

The data warehouse plays a pivotal role as a centralized storage facility for managing extensive amounts of structured and semi-structured data. Its purpose is to facilitate efficient analysis, reporting ,and decision-making processes [5].
In this context, NoSQL systems have emerged as superior alternatives to relational systems due to their capabilities in handling vast volumes of data and providing enhanced flexibility. Consequently, researchers have made significant contributions by proposing diverse methodologies to transform the multidimensional conceptual model into a NoSQL (schema-less) model [6].
Mainly for document-type databases like MongoDB. As evidenced, in recent research the use of NoSQL systems for implementing data warehousing and decision support systems has been investigated. Several studies have proposed mapping rules and transformation approaches to automatically translate multidimensional conceptual models into NoSQL logical models particularly focusing on column-oriented and document-oriented models.

Chevalier et al [3] conducted research on using NoSQL systems for implementing OLAP systems. They proposed a set of mapping rules to automatically translate multidimensional conceptual models into NoSQL logical models, specifically column-oriented and document-oriented models. The motivation behind their investigation is the flexibility and scalability advantages offered by NoSQL

systems. The research also explored data loading issues and the precomputation of data aggregates. A case study on RSS feeds was presented to illustrate the concepts, and a comparison with traditional relational database implementations was conducted.
The research paper contributes to understanding the implementation of multidimensional data warehouses and highlights the importance of mapping rules in successful integration with data sources.
In a related study *Chevalier et al* [3], explored the potential of document-oriented NoSQL systems for multidimensional data warehousing. They proposed two document models equivalent to normalized and denormalized data storage and introduced extended OLAP cuboids using nesting and arrays. The advantages of the document-oriented models were compared to classical relational models and experimental results showcased their performance differences.

Another research paper by *Bouaziz, Nabli and Gargouri* [4] focused on designing a data warehouse schema from NoSQL databases specifically document-oriented databases. The paper discussed the challenges posed by the increasing volume and variety of data, particularly unstructured data in the context of Big Data. The authors highlighted the limitations of traditional relational databases and introduced NoSQL systems as an alternative for managing and processing Big Data. The paper explored the extraction of schema and structure identification from document-oriented databases and discussed multidimensional concepts and the modeling schema of the data warehouse.
Additionally, the paper *"Towards Schema-independent Querying on Document Data Stores"*[7] addressed the challenge of querying structural heterogeneity in document-oriented databases. The easyQ approach was proposed enabling schema independent querying for multi-structured documents through virtual integration and a data dictionary. This approach simplified query formulation and accommodated the evolving structural heterogeneity in document-oriented databases.

Furthermore, the research paper [8] focused on the extraction of conceptual models from NoSQL databases specifically Mongodb to conceptual model approach based on model-driven architecture MDA. Provided a precise and automatic method for reverse engineering NoSQL databases and extracting the conceptual model. The paper emphasized the handling of links between tables in the extraction process.

In summary, the existing research in document-oriented data warehousing has contributed to understanding the implementation of multidimensional data warehouses transformation rules schema design and querying in NoSQL systems. The studies have explored the advantages of NoSQL models such as flexibility scalability and improved performance. However, research is needed to optimize performance and compare different data modeling choices in document-oriented data warehouses.

By examining the existing body of work, we found that previous research efforts have primarily focused on individual aspects of document-oriented data warehousing. A comprehensive comparison of the performance between the embedded and reference document approaches has been conspicuously absent, though.
Therefore, the key contribution of our research paper lies in addressing this research gap by conducting a rigorous performance evaluation of the two concepts , embedded and reference document approaches. To achieve that, we start with formally describing the source (multidimensional model) and the target (documents- oriented models.

3 Formal representation of the multidimensional schema

The conceptual modeling phase serves as an indispensable foundation for data warehousing initiatives. Within this phase data warehouses are structured and designed in a multidimensional manner utilizing key concepts such as facts measures dimensions and hierarchies. These fundamental elements of multidimensional modeling provide a structured framework to organize and analyze data effectively.
To formalize the process, as outlined in reference [7][9]. We adhere to the following principles and guidelines:

A Multidimensional snowflake schema namely MSS is defined by $(F^{MSS}, D^{MSS}, SFunc^{MSS})$ where:

Fact(F^{MSS}): contains the numerical data that is being analyzed, such as Loans(the illustrated example below). Which is ***one and only one fact*** in the snowflake schema. Formally, a fact is defined by (N^F, M^F) where:

- (N^F) is the name of the fact.
- (M^F) is a set of measures, each associated with an aggregation function.

Dimensions (D^{MSS}) that contain information about the characteristics of the data, such as time, books, and borrowers.
The dimensions can be further normalized into sub-dimensions to provide more granular details.

Formally, A dimension is denoted $D_i \in D^{MSS}$ (abusively noted as D), is defined by (N^D, A^D, H^D) where:

- N^D is the name of the dimension
- A^D=a^{D_1}, a^{D_2} ... , a^{D_n} is a set of strong and weak dimension attributes
- H^D =H^{D_1},H^{D_2} ..., H^{D_n} is a set of hierarchies.

And a **Hierarchy** which can be used to organize data By putting similar items in one group. A hierarchy consists of levels, and each level may have multiple attributes.
Hierarchies can help to streamline data analysis and make it simpler to spot

patterns and connections in the data.

$SFunc^{MSS}$ is a function that associates fact F^{MSS} to sets of dimensions D^{MSS} along which it can be analyzed (link Fact-Dimension).

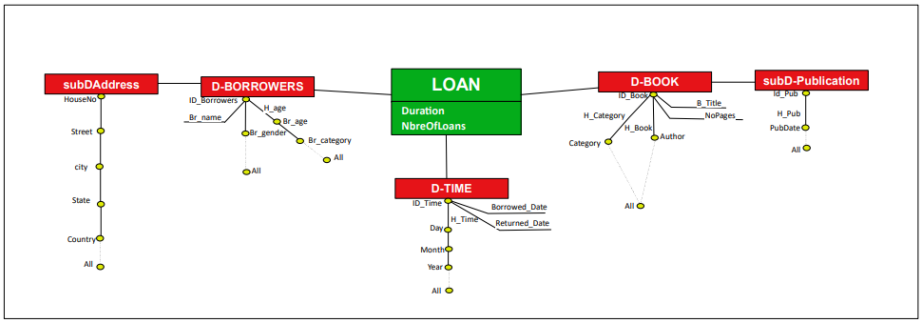

Fig. 1. Multidimensional conceptual snowflake schema representation.

4 Formal Representation of Document-oriented Databases

Formally, a **NoSQL document-oriented database** can be defined as a collection C composed of a set of documents D_i [3]:

$$C = \{D_1, D_2, \ldots, D_n\}$$

Each Document D is defined by a set of pairs$D_i = (\boldsymbol{Att}_i^1, V_j^1), \ldots, (\boldsymbol{Att}_i^m, V_j^m)$, $j \in [1, m]$, where $\mathbf{Att}_j^i$ is an attribute (which is similar to a key) and V_j^i is a value that can take two forms[3] :

- The value can be atomic: Simple attribute.
- The value itself is a collection of nested documents, each with its own set of attributes and values: Compound attribute.

The formal descriptions mentioned above are essential in helping us define and automate the transformation rules for converting conceptual models into NoSQL representations.

The next section aims to propose transformation rules for NoSQL data warehouses (DWs). As a reminder a DW schema consists of a fact with measures and a set of dimensions with attributes. Our goal is to map the dimensions based on their attributes and the fact based on its measures.

5 Mapping rules

It's important to note that this is a general process and the specific mapping rules may vary depending on the data, the requirements of the data warehouse, and the specific implementation of MongoDB.

In this context, we define two types of transformation rules for NoSQL DWs. One uses the concept of embedded documents, and the other focuses on using references. The first transformation approach involves utilizing embedded documents where the attributes of sub-dimensions and dimensions are directly embedded within the fact document by applying the transformation rules we establish guidelines for mapping the attributes of dimensions sub-dimensions and dimensions to the appropriate fields within the fact document.
While the second transformation approach revolves around using references. In this case, the fact document contains references to the corresponding dimension documents, and the dimensions documents contain references to the sub-dimensions documents. The transformation rules guide a proper mapping from Multidimensional schema to document-oriented models.

based on the formalized structures of both the multidimensional and document oriented schemes, we define two groups of transformation rules

Transformation rules From $MSS(F^{MSS}, D^{MSS}, SFunc^{MSS})$ to Embedded-document schema

1.1 **Transformation rules: 1^{st} Contribution:**
- **Rule 1:** Each conceptual multidimensional schema $(F^{MSS}, D^{MSS}, SFunc^{MSS})$ is translated in a collection $C = \{D_1, D_2, \ldots, D_n\}$.
- **Rule 2:** Each Fact $F^{MSS}(N^F, M^F)$ is translated in a compound attribute Att^{CF} where: Each measure m_i is translated into a simple attribute Att^{smp}
- **Rule 3:** Each *dimension* $D_i \in MSS(F)$ is converted into a compound attribute Att^{CF} (a **nested document**). Each *sub-dimension* $subD_i \in MSS(F)$ is converted into a compound attribute Att^{CF} contained in the D_i.
- **Rule 4:** Each attribute $Att_i \in Att_D$(parameters and weak attributes) of the dimension D_i is converted into a simple attribute.

Transformation From $MSS(F^{MSS}, D^{MSS}, SFunc^{MSS})$ to Reference-document schema

1.2 **Transformation rules: 2^{nd} Contribution:**
- **Rule1** : Each conceptual snowflake schema $(F^{MSS}, D^{MSS}, SFunc^{MSS})$, (one F, its dimensions, and its subdimensions) are translated in several collections $C = \{C_1, ..., C_n\}$.
- **Rule 2 :** The fact $F^{MSS}(N^F, M^F)$ is translated into a collection C^F.

- **Rule 3 :** Each measure m_i is translated into a simple attribute Att^{smp}, including the foreign keys to the related dimension as references.
- Rule 4: Each dimension Dim_i is converted into a collection $C_i^{Dim_i}$ including the foreign keys to the related sub-dimensions $subDim_i$ collections as references.
- **Rule 5 :** Each sub-dimension $subDim_i$ is converted to a collection $C_i^{subDim_i}$ as well.

2 Use MongoDB's built-in aggregation pipeline to perform complex queries.

Remark: From a perspective it is possible to combine the embedded and reference approaches employing a query driven methodology.

6 Experiments and Discussion

To validate our work we conducted two experiments based on the mapping rules mentioned above in these experiments. We built two data warehouses each employing a different approach for organizing and storing dimensions and sub dimensions.

In the first data warehouse (**1st DWH** : Using **embedded documents**)all dimensions and sub dimensions are combined into one collection namely the fact collection. This approach follows the document embedded approach where the attributes of dimensions and sub-dimensions are embedded within the fact document.

In the second data warehouse (**2nd DWH**: Using **referenced documents**) the fact dimensions and sub dimensions are separated into different collections. These collections are linked using *references* allowing for more flexibility and modularity in the data organization.

During our experiments we explored various data volumes small medium and large containing 10.000 , 20.000, and 50.000 collections respectively.
To analyze this data comprehensively we employed Mongo DB's powerful aggregation pipeline. This versatile tool, allowed us to run complex queries such as roll up, drill down, joins and data filtering.
By applying the same three different queries to both data warehouses we obtained identical results albeit with varying execution times. To provide a clearer understanding of the three queries, we will describe an illustrated example.

Illustrated example: Based on the above formalization, the multidimensional snowflake schema MSS illustrated by the **Fig. 1** will be defined as following:

-$F^{MSS} = \{F^{Loan}\}$,

- D^{MSS} = {$D^{Borrowers}$, D^{Time}, D^{Book} },

- $SubD^{MSS}$={ $D^{Publication}$,$D^{Address}$ } and

- $SFunc^{MSS}(F^{Loan}) = \{D^{Borrowers}, D^{Time}, D^{Book}\}$.

- The fact F^{Loan} is defined by *(Duration, {SUM(NbreOfLoans)})* and it is analyzed according to five dimensions, each consisting of several hierarchical levels (called detail levels or parameters)

- The Book dimension (D^{Book}) with parameters: *ID_ Book* (along with the weak attribute *B_ Title and NoPages), Author, and Category parameters,* organized using two hierarchies, ***H_ Book*** *and* ***H_ Category****:*

H_ Book*=({ID_ Book {B_ Title,NoPages}, Author, All});* ***H_ Category****={ID_ Book, Category}*

- Followed by a subdimension called *subD_ Publication* which has two attributes *Id_ Pub, and PubDate,* and is organized by a ***H_ Pub*** *hierarchy.*
H_ Pub={Id_Pub, PubDate}

- The Borrowers dimension ($D^{Borrowers}$) with parameters *ID_ Borrowers* (with weak attributes Borrowers name *Br_ name*), *Br_ age*, *Br_ Category*, and *Br_ gender* organized using two hierarchies ***H_ Age*** and ***H_ gender***.
H_ Age={*Br_ age, Br_ category*}; ***H_ gender***={*ID_ Borrowers, All*}

- Followed as well by a *sub-dimensio*n called *subDAddress* which has five parameters (*HouseNo, Street, city, State, and Country*) and is organized by a ***H_ Address*** hierarchy. *H_ Address={IdAddress, All}*

- The *Time dimension* (D^{Time}) with parameters *Day, Month, and Year.*

In this section, we will explore the three queries denoted as Q1, Q2, and Q3 . These queries will be examined and analyzed in detail in the upcoming section, to gain a deeper understanding of their characteristics and performance:

Query 1: Lists borrower distribution by gender using MongoDB aggregation. Utilizes *group* and *sum* to count documents per gender.
Query 2: Lists documents distribution among publishers. By focusing on counting records per publisher, enabling an understanding of loan issuance across different publishers.
Query 3: displays documents grouped by the 'Returned_date' field and calculates the total count within each group. This process provides insights into the overall frequency of returned items on different dates.

Experiments description

– **1st experience**

In the first experiment we created a collection that combines the fact dimensions and sub dimensions together. In this collection the measurement attributes $Att_{NbreOfLoans}$ and $Att_{duration}$ are treated as atomic attributes. The collection is structured as follows Borrowers', Book and time are nested documents within the fact collection. Additionally, the sub-dimensions publications and address are nested documents within the Book and Borrowers documents respectively. This hierarchical structure is depicted in the figure .
By consolidating all the relevant information into one collection we can efficiently store and retrieve the data for analysis and querying purposes.
The document embedded approach allows for a compact representation of the data reducing the need for multiple collections or complex references.

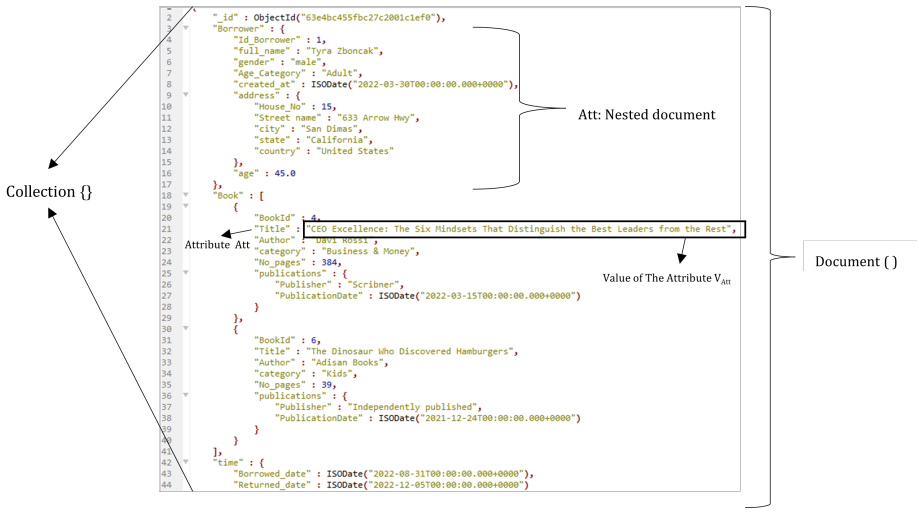

Fig. 2. The hierarchical structure of the fact collection.

– **2nd experience**

In the second experiment we created three collections to represent the fact dimensions and sub-dimensions. Each collection is responsible for storing specific data related to its corresponding entity. In this approach the fact collection is linked to the dimension and sub-dimension collections using reference documents. These references establish the relationships between the fact data and the corresponding dimensions and sub dimensions

Discussion:

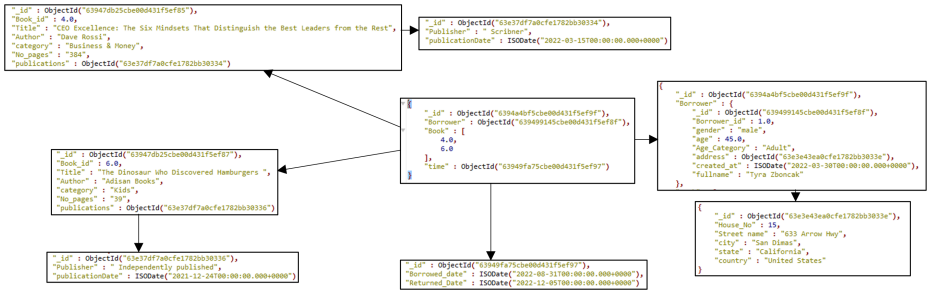

Fig. 3. The hierarchical structure of the 2nd DW.

Comparing Data Warehousing Approaches: Embedded vs. Referenced Documents Approach: Analyzing the results of the experiment comparing the performance of a NoSQL data warehouse using an embedded document and a reference document approach, while varying the data volume.
We present the findings in Fig.4., the metric used to evaluate performance is the execution time of the queries. Fig.4. displays the execution time comparison between the first data warehouse (1st DW) utilizing the embedded document approach and the second data warehouse (2nd DW) employing the reference document approach. The table presents the execution time in milliseconds (MS) for each query.

Queries	1ST DWH			2ND DWH		
	10k	20K	50K	10K	20K	50K
Q1	12ms	22ms	70ms	554ms	2349ms	>60000ms
Q2	10ms	18ms	53ms	>60000ms	*	>60000ms
Q3	17ms	17ms	72ms	621ms	>60000ms	>60000ms

*Execution Interrupted (A Highly Significant Value)
Exceeded Limited Execution Time: > 60000 ms.

Fig. 4. Comparing Execution Time (ms) Between the 1st and 2nd Data Warehouses.

Discussing and analyzing the results of the experiment comparing the performance of a NoSQL data warehouse using an embedded document and a reference document approach.
In the initial comparison between the first data warehouse and the second data warehouse with a dataset of 10.000 collections. A noticeable difference in query execution time results is observed between the 1st and 2nd DWH.
In second comparison using 20.000 collections, several noteworthy observations emerge in the case of the 2nd DWH, as well : There is a notable increase in execution time values this is primarily attributed to the extensive lookups per-

formed between different collections compared to the 1st DWH.
For the second query, an interruption occurred in the 2nd DWH, during the execution timeframe. This interruption likely impacted the completion of the query. Additionally, within the context of the 2nd DWH, the execution time for the third query surpassed the predefined limit for execution time. These observations underscore the significance of efficient data retrieval and processing strategies within the Referenced document approach, particularly when dealing with complex lookups and query execution times.
In the third experiment utilizing a dataset of 50,000 collections, a distinct contrast becomes evident between the embedded and referenced data warehousing (DWH) approaches. Notably, the execution time for the embedded document method is remarkably brief in comparison to the reference DWH. It's noteworthy that all results originating from the reference documents surpass the 60,000ms threshold. This disparity can be attributed to the utilization of lookups and the substantial data volume associated with the reference approach.

Data volume's impact on Execution time: Fig. 5. displays the execution time comparison between the first data warehouse (1st DW) utilizing the embedded document approach and the second data warehouse (2nd DW) employing the reference document approach.

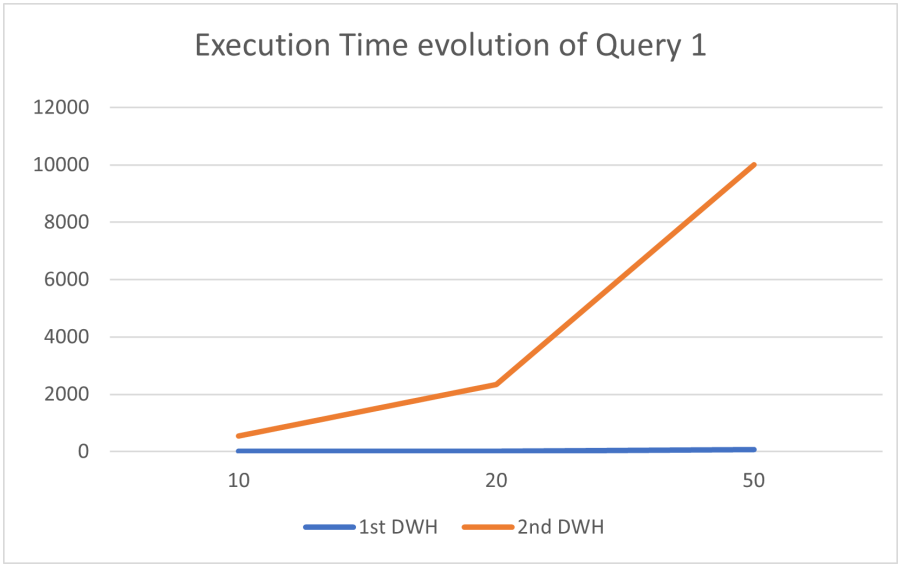

Fig. 5. Representative Performance diagram

By applying query Q1 to collections above in both the embedded data warehouse (1st DWH) and the referenced data warehouse (2nd DWH), and by systematically varying the data volume across three distinct collection sizes 10.000, 20.000, and 50.000 entries in both data warehouses. we gain valuable insights into how data volume directly influences execution time.

In the embedded DWH (1st DWH), the execution times remain relatively consistent and significantly shorter across the different collection sizes. The execution times of 12ms, 22ms, and 70ms for the 10k, 20k, and 50k collections respectively showcase the efficiency of this approach.
Conversely, the referenced DWH (2nd DWH) exhibits a notable rise in execution times as data volume grows. The execution times of 554ms, 2349ms, and 10000ms for the 10k, 20k, and 50k collections respectively indicate that the referenced approach is more likely to see a noticeable increase in execution times as the data volume gets larger.

This diligent work into the correlation between data volume and execution time, not only provides valuable insights for optimizing data warehouse performance, but also contributes to a better understanding of how a data management works.
This divergence in execution times between the two approaches underscores a crucial point: the embedded approach proves a capacity to manage and process larger volumes of data without a significant increase in execution time. Meanwhile, the referenced approach encounters challenges in maintaining efficient execution times as the data volume expands.

7 Conclusions

In this paper we have proposed transformation rules that ensure the successful translation from conceptual DW schema to logical NoSQL model through document embedding.
The experiments conducted shown improved query response times for document-embedded data warehouses. These findings are useful for the design and implementation of data warehouse systems based on NoSQL document-oriented systems such as Mongodb. Moving forward, there are key topics for future investigation, one of which is resolving the issue of data duplication, which can have an influence on storage efficiency and database size in document embedded data warehouses.

Acknowledgements We would like to express our sincere gratitude to Professor Faiza Ghozzi for her invaluable guidance and support throughout this research project. Her expertise and insightful feedback greatly contributed to the quality of this work. Additionally, we extend our thanks to the anonymous

reviewers for their constructive comments and suggestions, which helped us improve the manuscript.

References

1. Rocha, L., Fernandes, L., Pereira, J., Bernardino, J.: A framework for migrating relational datasets to NoSQL. *Procedia Computer Science*, **51**, 2593–2602 (2015)
2. Sellami, A., Nabli, A., Gargouri, F.: Transformation of data warehouse schema to NoSQL graph database. In: *Proceedings of the 3rd International Conference on Big Data, Cloud and Applications*, pp. 1–6 (2018). IEEE
3. Chevalier, M., Tournier, R., Boussaid, O., Gruenwald, L.: Implementing multidimensional data warehouses into NoSQL. In: *Proceedings of the 17th International Conference on Enterprise Information Systems*, pp. 172–183 (2015). SciTePress
4. Bouaziz, S., Nabli, A., Gargouri, F.: Design a data warehouse schema from document-oriented database. *Procedia Computer Science*, **159**, 221–230 (2019)
5. Bicevska, Z., Oditis, I.: Towards NoSQL-based Data Warehouse Solutions. *Procedia Computer Science*, **104**, 104-111 (2017). https://doi.org/10.1016/j.procs.2017.01.080
6. F. Abdelhedi, A. Ait Brahim, F. Atigui, and G. Zurfluh, "UMLtoNoSQL: Automatic Transformation of Conceptual Schema to NoSQL Databases," in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 2017, pp. 272-279, doi: 10.1109/AICCSA.2017.76.
7. Hamadou, B., Ghozzi, F., Péninou, A., Teste, O.: Towards Schema-independent Querying on Document Data Stores. (2018).
8. Brahim, A. A., Ferhat, R. T., Zurfluh, G.: Extraction process of conceptual model from a document-oriented NoSQL database. In: Proceedings of the 11th International Conference on Knowledge and Systems Engineering (KSE 2019), Oct 2019, Da Nang, Vietnam, pp. 1-5.
9. Nabli, A., Feki, J., Gargouri, F.: Automatic construction of multidimensional schema from OLAP requirements. In: Proceedings of the International Conference on Advanced Intelligent Computing and Systems (AICCSA), (2005), p. 28.