Computing and Informatics, Vol. 41, 2022, 813–833, doi: 10.31577/cai_2022_3_813

UML4NOSQL: A NOVEL APPROACH FOR MODELING NOSQL DOCUMENT-ORIENTED DATABASES BASED ON UML

Mohammed ElHabib Maicha, Youcef Ouinten, Benameur Ziani

LIM laboratory
Amar Telidji University
Laghouat, Algeria
e-mail: {mh.maicha, ouinteny, bziani}@lagh-univ.dz

Abstract. The adoption of Big Data systems by the companies is relatively new, although the data modeling and system design are ages old. Despite the fact that traditional databases are built on solid foundations, they cannot handle the swift and massive flow of data coming from multiple different sources. Herein, NoSQL databases are an inevitable alternative. However, these systems are schemaless compared to traditional databases. It is important to emphasize that schemaless does not mean no-schema which would mean that NoSQL databases do not need modeling. Hence, there is a need for conceptual models to define the data structure in these databases. This paper sheds a light on the importance of the UML in showing how to store Big Data described through meta-models within NoSQL databases. We propose a novel Big Data modeling methodology for NoSQL databases called UML4NoSQL, which is independent of the target system, and taking into account the four Big Data characteristics: Variety, Volume, Velocity, and Veracity (4 V's). The approach relies on the UML blocks with a data-up technique; it starts with a use-case and the class diagram resulting from the understanding of the data at hand and the definition of the developer's strategies while focusing on the user's needs. To illustrate our approach, we take a case study from health care domain. We show that our approach produces designs that can be implemented on NoSQL document-oriented system with respect to Big Data 4 V's.

Keywords: Big Data, UML, database modelling, NoSQL, document-store, UML4NoSQL

1 INTRODUCTION

The design phase plays a vital role in the life cycle of any computer system. In order to reach a valid model, a variety of methods are available. Among these, UML is the one that is extensively used. It has already proven its worth as a highly effective language of modeling traditional systems such as web applications and even complex systems arising from the increasingly complex needs of contemporary companies. However, nowadays companies often deal with a huge amount of data that traditional databases cannot support. Thus, NoSQL databases have become a popular alternative due to their capability of handling Big Data. Consequently, the future of UML in the era of Big Data has become a relevant and debatable issue.

To deal with this issue, we will start by stating what the professionals say about Big Data modeling. Adamson, the president of information management consultancy of Oakton Software argues that "There's a lot of confusion right now in the market ... that leads people to believe you don't need a model with NoSQL technologies"[1]. The issue of compatibility of Big Data and conceptual models was also raised in the ER 2016, the 35th International Conference on Conceptual Modeling.

On the other hand, the definition of Big Data in the literature fits the definition of a computer system with major differences at four levels:

1. The variety of data sources which leads to different forms of data.
2. The volume with massive growth in the scale of data quantities that reach almost incomprehensible proportions.
3. The velocity which refers to data generated at high speed by sensors or multiple events and need to be processed in near/real time.
4. The veracity which includes two aspects: data consistency and data trustworthiness.

For several years remarkable efforts such as in [1, 2, 3, 4, 5, 6] and [7] have been devoted to the study of Big Data and a consensus is emerging that Big Data is a computer system. Therefore, as any other systems it requires modeling so that its boundaries and uses can be identified. Works about NoSQL database design using UML, including the one presented by Shin et al. [8], are very scarce. Overall as reported by Mior et al. [9], most of the existing works dealing with NoSQL databases do not give much importance to the design methods. This is because so far, the use of NoSQL databases is directed only to solving specific application problems of non-functional requirements like performance, availability, and scalability, while relational databases which have been studied for a long time, have design methods to implement databases from data requirements. One way out of this is to investigate the adaptation of UML so as to provide NoSQL technology with appropriate design methods.

[1] TechnoPedia Website, October 2020

If we do accept the fact that UML can be used to design NoSQL databases then several practical issues arise:

1. It is crucial to identify the possibility of transforming UML diagrams in such a way that they can be used in the NoSQL design process.
2. The process must provide comprehensive techniques and guidelines for effective data modeling methodology for NoSQL databases.
3. The process should finally lead to a complete and valid physical data model to be implemented in NoSQL systems.

To deal with all these issues, we present in this paper "UML4NoSQL", an original UML-based approach which adapts the most descriptive UML blocks so as they yield a metamodel describing Big Data which can then lead to several NoSQL physical models.

A high-level overview of our methodology is illustrated in Figure 1. The UML's basic building blocks and application workflow are used to start the conceptual model. We then introduce a logical level that describes data according to the common features of the document-oriented database. Additional physical optimization concerning data types, and ordering are then applied to generate a physical data model that can be instantiated in the desired NoSQL technology such as MongoDB.

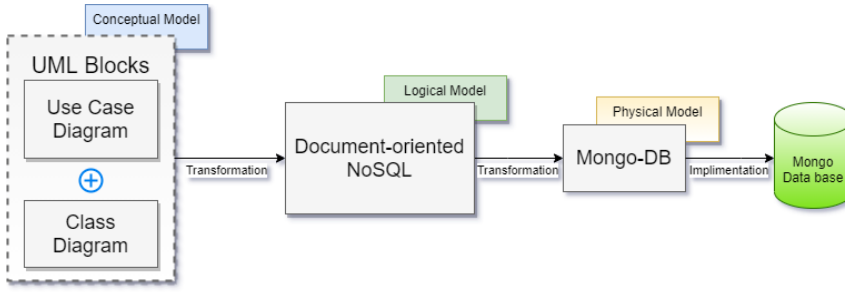

Figure 1. Overview of the proposed methodology

The remainder of this paper is organized as follows: Section 2 is devoted to a background and previous works on Conceptual Modeling in Big Data context. An illustrative example to state the problem is presented in Section 3, while Section 4 introduces our approach UML4NoSQL which goes through four steps:

1. Understanding DATA,
2. defining the users' needs,
3. the developers' overview,
4. generation of NoSQL Meta-Model.

Our approach aims to transform UML conceptual models into document-oriented NoSQL physical models. Section 5 draws conclusions as well as some perspectives for future work.

2 BACKGROUND AND RELATED WORK

2.1 Background

In this section we will cover the background necessary to the comprehension of the remainder of this paper. There are four main categories of NoSQL databases, each with different levels of scalability, flexibility, complexity, and functionality:

Key-Value Store: Key-Value Store databases store data in a schemaless way, where data is saved into an associative array and represented as a collection of key values pairs, assembled by a unique key and the rest of the data (values). (e.g., Oracle NoSQL and Redis).

Column Store: A Column Store database (also known as columnar databases), was motivated by Google's BigTable [10]. Instead of storing data in rows, data tables are stored as sections of columns of data. It is an extension of Key-Value Store database where columns can have a complex structure, rather than a blob value. Examples of this kind are Apache HBase or Apache Cassandra [11].

Document Store: This kind of databases are very similar to key value, where the value is a document, such as JSON, BSON, etc. Each document has a unique key which is assigned to retrieve the document. Document stores are good for semi structured data, they support aggregates and denormalized structures. Document stores are found in MongoDB, CouchDB, and others [12, 13].

Graph Database: Graph databases are based upon graph theory (set of nodes, edges, and properties). They are useful for inter connected relationship data such as social networks, and biographical interactions. The data is stored in nodes and relationships are represented as arrows connecting nodes between each other. It may, more accurately, be described as non-relational rather than NoSQL. Examples are Neo4j, Titan, and OrientDB [14].

The production of a conceptual model for any given NoSQL system based on UML requires three major elements: the UML's basic building blocks, the rules that dictate how those building blocks may be put together, and some common mechanisms that apply throughout the UML. Let us define in this section the UML's basic building blocks:

Things: Things are the basic object-oriented building blocks of the UML. They fall into four kinds: structural, behavioral, grouping, and annotational things [15].

Relationships: There are four kinds of basic relational building blocks of the UML: dependency, association, generalization and realization. There are also variations on these four – such as refinement, trace, include, and extend [15].

Diagrams: A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and paths (relationships). The UML includes thirteen kinds of diagrams all described in detail in [15]. Things are the abstractions that are first-class citizens in a model, relationships tie these things together and diagrams group interesting collections of things [16].

2.2 Related Work

The literature on modeling Big Data shows a variety of approaches devoted to the development of methodologies and tools supporting NoSQL database design [17, 18, 9]. The publications available in the literature that investigated this issue from the point of view of meta-modeling are very scarce. Indeed, most of the works propose a solution to transform from traditional databases to a specific NoSQL models in a limited case of studies.

Papers	**Inputs (e.g. UML Blocks)**	**Outputs (NoSQL Systems)**	**Big Data Characteristics**
[9]	Conceptual schema and statistics	Column-store	–
[17]	Aggregate-oriented view of data	System-independent data model	Scalability and consistency
[19]	Star schemas and lattice (OLAP)	Column-oriented and document-oriented	–
[20]	OLAP cube (Data Warehouse)	Columnar NoSQL cubes	Volume and Variety
[21]	Multidimensional conceptual model	Column-oriented and document-oriented	–
[22]	UML Class diagram	Document-oriented	Variability
[23]	UML Class Diagram OCL constraints	Graph database	–
[24]	UML Class Diagram	Column, document and graph databases	Volume, Variety and Velocity
[25]	Meta-Model of UML Class Diagram	Column-store	–
[8]	UML Conceptual data model - Peter Chen	Document-oriented	–

Table 1. A comparative study of NoSQL modeling approaches

Table 1 provides a comparison of all presented works according to how they convert the input model into a NoSQL output model. We also take the four features of Big Data as an additional metric.

As illustrated in Table 1, the authors in [19, 20, 21] have proposed a transformation process with a set of mapping rules from multidimensional models to column-oriented and document-oriented models. While the authors in [22, 23, 24]

took the UML class diagram as input to offer a columnar and document oriented models that respect, at most, two Big Data characteristics.

For graph-oriented databases, authors in [14] presented a comparative analysis issues and concepts on a list of graph database tools.

The most interesting approach to deal with this database family has been proposed by [25]. In this research work, the authors propose a framework that translates conceptual schema expressed using the UML into a graph representation.

Atzeni et al. [17] presented the first proposal of a general system independent approach to the design of NoSQL databases. The major drawback of this approach like many of related works, is that it relies only on the class diagram from the UML family. Authors in [8] have also found that current database design methods do not address non-functional requirements, they tend to refer to a preselected database; and they do not offer an evaluation process. To the best of our knowledge, studies on Big Data modeling, from a general perspective as we offer through this paper, do not offer a complete meta-models that can handle NoSQL databases independently of their technology. The present work is an attempt to deal with this issue.

3 RESEARCH MOTIVATION

To illustrate and motivate our work, we have chosen a case study from the health care field, where a NoSQL database have a legitimate role to play. The case study is about a 51-hospital system with 100 000-plus caregivers who deliver high-quality, cost-effective health care to millions of patients annually. Patient data resides in many systems, including electronic medical records (EMRs). The main objective on this health system is (1) to collect data about the decease development over time from patient charts, to improve patient care, treatment affordability and the health care experience, (2) to offer dashboards displaying detailed quality data and cost data. The dashboards allow practitioners and clinicians to see analytics that are related to every hospital, clinician and individual nursing unit, (3) to analyze and share data in a more organized way, making it easier for doctors to understand what behaviors positively or negatively impact patient care, which leads to substantial improvements in quality measures and large reductions in the cost of care.

Furthermore, the health system can integrate sensor data from patient-monitoring systems to improve alert predictability, it can also use weather and seasonal data to predict staffing and bed needs. This example can be considered as a Big Data problem, in the light of the "4 V" definition which are Volume, Variety, Velocity, and Veracity.

Volume. The vast amount of data collected over several years from all 51 hospitals can easily reach multiple terabytes.

Variety. Every patient has his own medical records come from diverse sources and, thus, in various shapes (e.g. structured, unstructured, semi-structured) and

formats (e.g. laboratory test results, blobs of text, medical reports, pictures, video, medical images, lists of medicines, etc.).

Velocity. The wide spreading of IoT and smart medical devices lead to the generation of real-time health data.

There are many tools in place to analyze patient data and advise health care professionals to take appropriate actions. For example, new wearable sensors can help tracking patient health trends that can be monitored by doctors. The collected data can be very helpful in monitoring the health of patients, ranging from blood pressure monitoring to other conditions right at home, although trying to respond to every health data stream can lead to inefficient and uneconomic use of resources. It is therefore important to decide which data requires immediate action and which can be deferred.

Veracity. Data quality includes integrated, reliable, complete, bias-free, and noise-free data. Hospitals aim at reducing the number of ER visits or Emergency visits of patients which increases health care costs and does not, necessarily lead to better outcomes for patients.

4 CONTRIBUTION

The objective of this study is to propose a new approach that improves the conceptual modeling process in Big Data environments. In the state-of-the-art solutions, most of authors rely solely on the class diagram to perform a transformation to the NoSQL model. However, the class diagram is not a unique affecting factor in the NoSQL model transformation process. In our approach we followed a data-up technique, which starts with a Use-Case and the Class diagram issued from the understanding of the data at hand and the definition of the developer's strategies while focusing on the user's needs. The main outcome of our proposal is a set of designs that can be implemented on a NoSQL Document-oriented system with respect to the Big Data 4 V's.

4.1 UML4NoSQL: A Bird's Eye View

In this paper we show how a UML use case diagram can be adapted to the Big Data. In a Big Data environment, the availability of data is not an issue; it is available at will. The most important issue is to know what to do with the data or what can be done with it. In our approach, we aim to deliver a good representation of the application data in a target NoSQL database (Document-oriented), and it is intended to support the foremost qualities of Big Data systems, well known as the 4 V's. As it can be seen from Figure 1, our methodology articulates around four major axes:

4.1.1 Understanding the Data

The first principle in a successful database design is to have a good knowledge of the data for which the design is intended by considering the most important criteria:

1. The type of data being analyzed,
2. The volume of data at hand and
3. How quickly do we need that data.

The authors in [26] and in [27] have widely investigated the challenges faced when dealing with Big Data management. These challenges include issues related to data quality, data streams, the dynamic evolvement of data, data heterogeneity and data modeling, multi-model databases, client and query interfaces, data compression, data encryption, access control and authorization.

From this perspective, we note that end users of Big Data are becoming increasingly non-technical. This leads us to the introduction of a new profile that we call *Citizen developer*.

4.1.2 Developer's Strategies and User's Needs

While the traditional UML diagrams maintain their focus on the system's needs in terms of functions, with the arrival of Big Data the diagrams will focus on the user's needs (usually they are decision-makers) in terms of treatments based on the data they have to deal with (Prescriptive, Predictive, Diagnostic and Descriptive analytics).

It is important to know what can be done with the data at hand in the short, medium and long term. Answers to this question can be modeled by a *Use Case diagram*, we apply the use case diagrams to visualize the behavior of a system, a subsystem, or a class. The use case diagram is a fundamental tool for identifying requirements [28]. This will allow users to comprehend how to use these elements, and the developers to implement them. Herein, we define two new blocks as follows:

Definition 1 (Super Container)**.** Hosts all the Big Data qualities required for a use case diagram which reduces the complexity of the diagram by avoiding the creation of multiple instances of a data service, as shown in Figure 2.

Definition 2 (Actors' Container)**.** Contains the group of actors who have the same behavior when querying the system. This will control the variability and the velocity of the actor's queries inside the container, as shown in Figure 3.

From this diagram we can decide what data to keep in our Big Data environment. Either we keep all the data or we keep only the corresponding data and eliminate the data that in the long term will become useless, by considering it as dirty data. This data will be modeled by an extended class diagram. In a parallel work, we aim at providing sequence diagrams to describe the scenarios of using Big Data, and activity diagrams for more detail on the used algorithms (e.g. data mining algorithms).

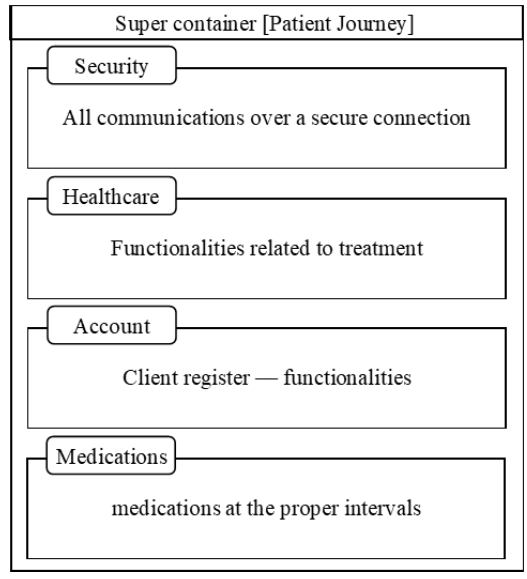

Figure 2. Super container

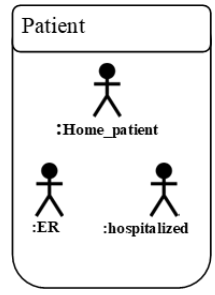

Figure 3. Actors' container

4.1.3 NoSQL Meta-Model

Several studies have indicated that Big Data is schemaless. At most we can say it is less-schema than other data. The main purpose of this paper is to draw attention to the importance of meta models for Big Data, because when accessing the data, we need a schema that helps to interpret it, so if there is no explicit model to use, we have to infer one. In fact, models are not static, fixed and complete artifact, rather a partial, dynamic and temporal view of the data to facilitate manipulating it at a specific instant. In traditional software development, the developers follow a top down approach [29], in which the model defines the data to use. We aim to propose

an approach that relies on a data-up technique, i.e., the used model is based on the data available.

4.1.4 Create Design That Scales Easily

An important key to successful database design is knowing the actual queries, queries have a large effect on schema design. In addition to ensure the correct support of the queries, we should take into account the access path of each query to organize data efficiently (denormalizing or using a relational schema). Normalized and denormalized databases are discussed in [22] and [18]. We concur with the authors [30, 11, 7], that NoSQL databases perform better when the data is denormalized. Rather than preserving a relational schema, it is better to denormalize the data so that we can take advantage of nested and repeated documents. Indeed, nested and repeated documents can maintain relationships without the performance impact of preserving a relational (normalized) schema.

4.2 UML4NoSQL: Proposal Details

In this section, we present guidelines for a successful switching from a class diagram (CDM) to an NoSQL document-oriented model (LDOM).

In Table 2, we summarize the correspondence between elements of UML class diagram and NoSQL data models. Later in the experimental phase, we apply the conversion process over the document-oriented schema.

	NoSQL Systems			
UML Class Diagram	**Key_Value**	**Column Store**	**Document Store**	**Graph Database**
Class	Associative array	Column	Collections	Graph
Attribute in Class	Key-value	name/value in column	Documents	Nodes
Association	Key-value pairs	Directory hierarchies	References / Embedded	Edges

Table 2. Correspondence between components of UML class diagram and NoSQL data models

To formalize our approach we introduce some definitions of some terms used in the building of the document-oriented model like Document, Collection and key-value (attribute, value).

Definition 3. Let D be a set of documents $D_j, j = 1, \ldots, m$. Each document is defined by a set of atomic or sub-documented couples (attribute, value):

$$\mathcal{D}_j = \{(Att, Val)_i; i = 1, \ldots, n\}, \quad j = 1, \ldots, m$$

where n is the number of attributes and m the number of documents. The couples

(attribute, value) of nested documents are sorted into a collection:

$$\mathcal{C}oll = \{(Att,\, Val)_i^j; j = 1, \ldots, m; i = 1, \ldots, n\}.$$

If $\mathcal{C}^A$ is a subset of $\mathcal{C}oll$, representing a class $\mathcal{A}$, we assume that:

1. $\exists D_i, D_j \in D$ such that $D_i \subset \mathcal{C}^A$ and $D_j \subset \mathcal{C}^B$,
2. $\mathcal{C}^{AB} = \{C^A\{C^B\}\}$ represent the fact that C^B is embedded in C^A.

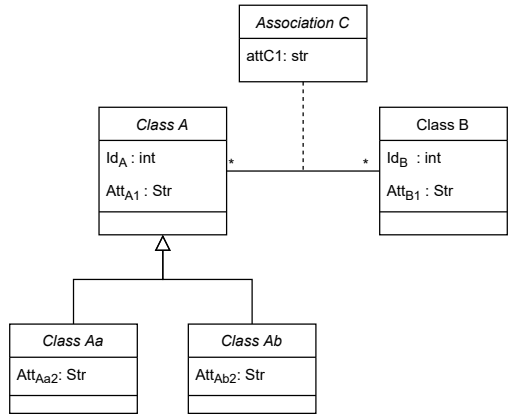

Figure 4. Passage from CDM to LDOM

The process of passage from CDM to LDOM, for the case illustrated in Figure 4, goes through the following steps:

- Each class is transformed into a collection consisting of a set of documents, where class attributes are transformed into attributes in each document:

$$C^{classA} \subset \mathcal{C}oll,$$
$$C^{classA} = \{(Att,\, Val)_{id}^A, (Att,\, Val)_{Att_{A1}}^A\}.$$

- A basic association (1..*) between the two collections, C^{classA} and C^{classB}, where the primary key of C^{classB} migrates to the collection C^{classA} as a foreign key, generates a nested collection N^{classB} which is C^{classB}:

$$C^{AB} = \{C^{ClassA}\{C^{ClassB}\}\} = \left\{ \begin{array}{l} (Att,\, Val)_{id}^A, (Att,\, Val)_{Att_{A1}}^A, (Att,\, Val)_{id}^B, \\ N^{classB} : \{(Att,\, Val)_{id}^B, (Att,\, Val)_{Att_{B1}}^B\}. \end{array} \right\}$$

- An association many to many (*..*) between two collections, C^{classA} and C^{classB}, creates a new collection $Coll^{AB}$ containing the attributes of the association class,

plus the primary keys of the two participating classes plus the two nested collections N^{classA} and N^{classB} with their own attributes.

$$Coll^{AB} = \left\{ \begin{array}{c} (Att,\, Val)^{A}_{id}, (Att,\, Val)^{B}_{id}, (Att,\, Val)_{AttC1}, \\ N^{classA} : \{(Att,\, Val)^{A}_{id}, (Att,\, Val)^{A}_{Att_{A1}}, \\ N^{classB} : \{(Att,\, Val)^{B}_{id}, (Att,\, Val)^{B}_{Att_{B1}}. \end{array} \right\}$$

- The DOM will be made up of the parent-child sets (Heritage association). In each child nested collection $N^{classAa}$ and $N^{classAb}$, we find the no-key attributes of the class plus the primary key composed of the parent-child class keys. The "Sort" value indicates which child class should be used to complete the information in the generated collection C^{AaAb}.

$$C^{AaAb} = \left\{ \begin{array}{c} (Att,\, Val)^{A}_{id}, (Att,\, Val)^{A}_{A1}, (Sort, \\ N^{classAa} : \{(Att,\, Val)^{Aa}_{id}, (Att,\, Val)^{A}_{id}, (Att,\, Val)^{Aa}_{Att_{Aa2}}\}, \\ N^{classAb} : \{(Att,\, Val)^{Ab}_{id}, (Att,\, Val)^{A}_{id}, (Att,\, Val)^{Ab}_{Att_{Ab2}}\}). \end{array} \right\}$$

5 ILLUSTRATIVE EXAMPLE

The evaluation in this case is not an easy task, as stated by Roy-Hubara et al. in [31]. To demonstrate the applicability and the suitability of our approach we choose to illustrate its use in this section through an illustrative example. Nevertheless, some research efforts should be made to find answers to the issue of evaluation so as the performance of NoSQL datatbase designs become measurable and comparable.

5.1 Description of the Experimental Case

Before proceeding to the modeling of the use case, we create an UML logical data model independent from any database model. This model is then compared with the model resulting from our approach.

The field of health care has already taken great advantages from Big Data. We have therefore chosen a case study from this field to validate our approach. Figure 5 shows a fragment of LDM for the health care analytic system described in Section 3.

5.2 Applying the UML4NoSQL Approach

In this section we apply our approach on the above running example by describing the two following steps:

1. Modeling developer's strategies and users' needs,
2. Generated NoSQL data model.

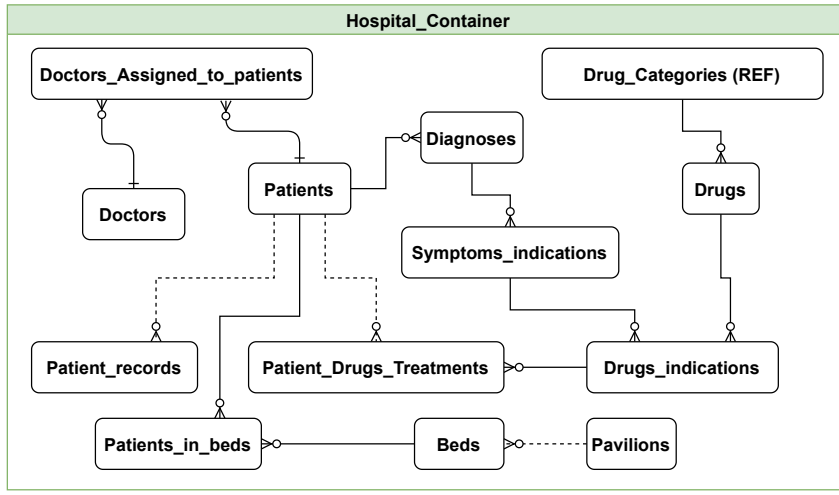

Figure 5. Logical data model for health care system

Modeling developer's strategies and users' needs. In order to establish the context of the system, we start by identifying the Actors that surround it and the interaction between them. Applying the first and the second UML4NoSQL data modeling aspects leads to the diagrams depicted in Figures 6, 7 and 8.

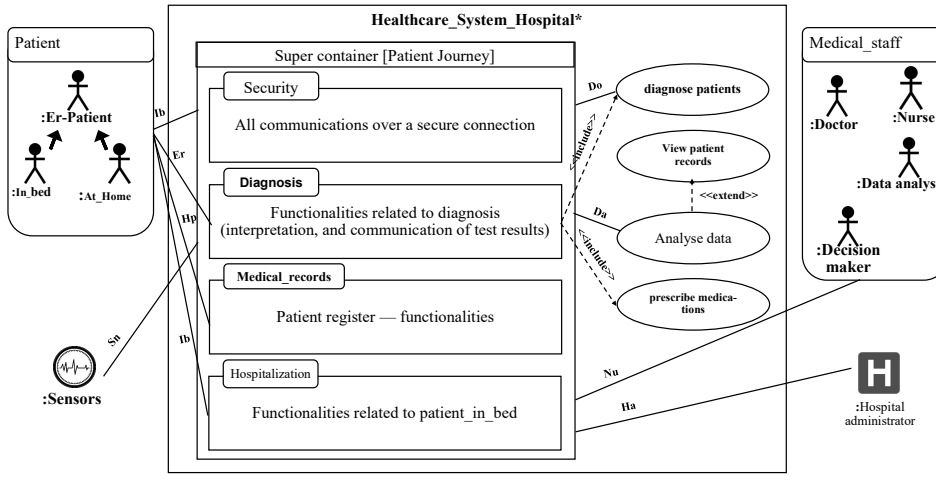

Figure 6. Enhanced use case diagrams for health care system

The proposed enhanced use case diagram introduces a super container, as can be seen in Figure 6. The super container hosts all the Big Data qualities required for a use case diagram and reduces the complexity of the diagram by avoiding the

creation of multiple instances of a data service. For example, the *Patient journey* super container will allow to monitor the process of a patient diagnosis accomplished by a doctor. In order to control the variability and the velocity of the actor's queries, we also propose the use of a container for a group of actors who have the same behavior when querying the system.

We now describe how the communication lines between the patient journey super container and the different actors' containers are represented to connect them with other sub use cases. The scenario we describe here is that the Data analyst (actor) performs analytics on a sample of the targeted Big Data in order to understand the patient's health behavior changes. All controls within the super container are applied, because the patient journey super container is on the communication line between the Data analyst and the sub use case. As there is more than one actor, the communication lines must be labeled. By referring to the same scenario mentioned above, the connection between the Data analyst and the container is labeled with "**Da**".

Now, when we have developed the overall framework of the system through a Use Case diagram, we come to identify two major steps to understand: the to-be-managed data, and how a data-driven application needs to access such data. The first being captured via a conceptual data model presented in Figure 7.

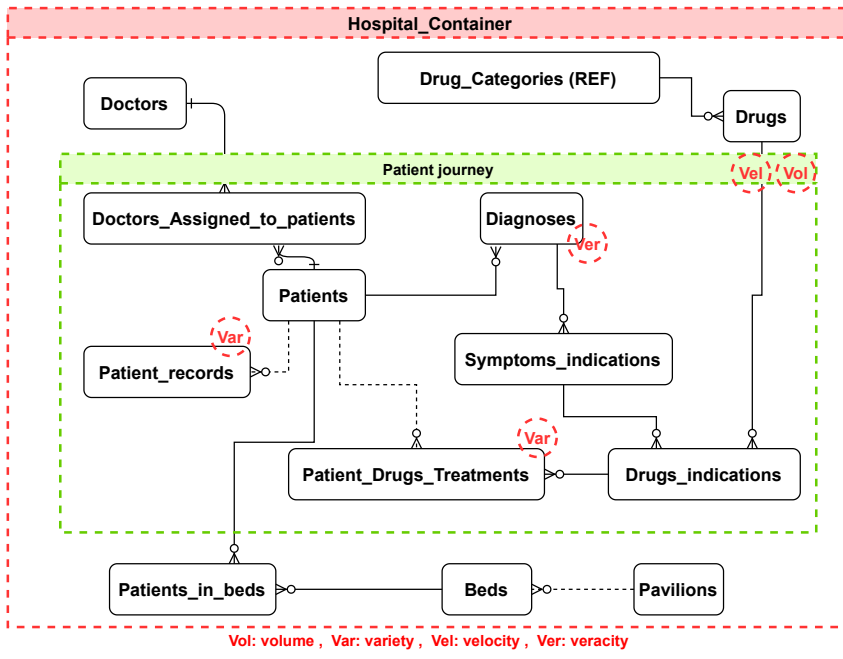

Figure 7. An enhanced conceptual data model for the health care use case

This data model is enriched with information regarding to the 4 V's. For instance, the developer has to anticipate the data volume (Vol) by assuming information on the number of occurrences produced by entity-relationships, as well as its growth velocity (Vel). Depending on the detail of representation of data types, the variety (Var) is linked to attributes. The veracity (Ver) is represented depending on the confidence of the developer in the data sources, at two levels:

1. the attribute; when characterizing trustworthiness in value quality, and
2. the aggregate; when characterizing the relationship between entities.

Every access pattern specifies what attribute(s) to search for/on, to order by, or to aggregate on.

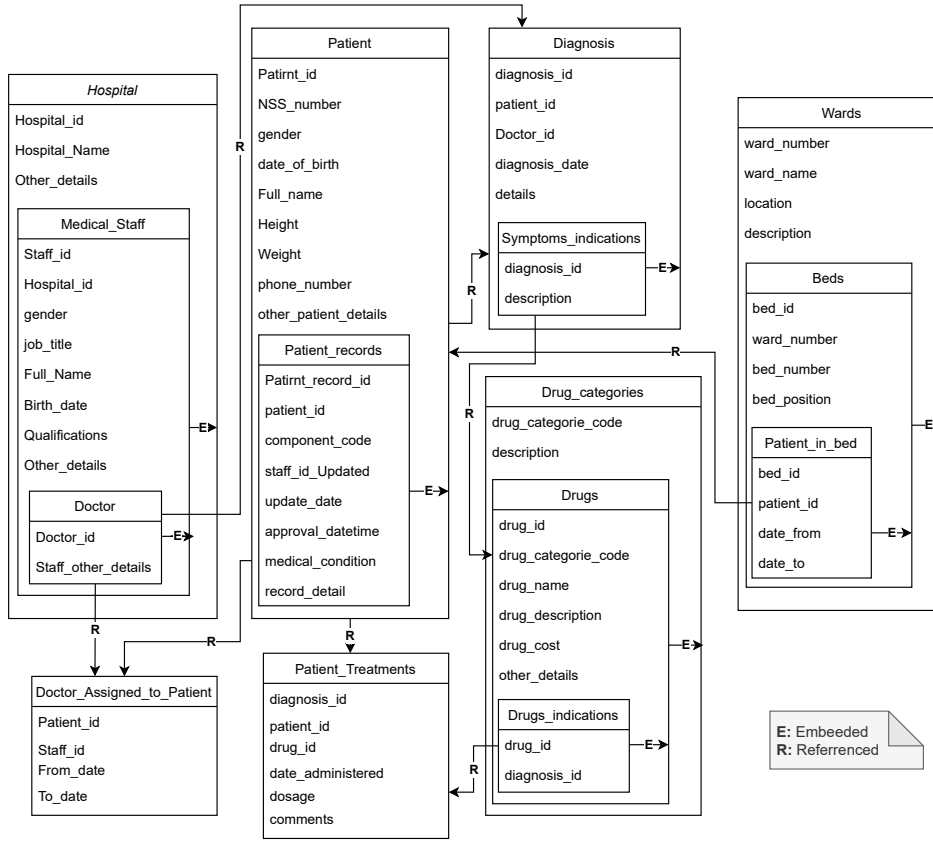

Figure 8. Optimized logical document-oriented model

Generated NoSQL data model. We perform data modeling principles seen in Section 4.2, to represent a query-driven transition from an enhanced conceptual data model to a logical data model (Figure 8).

Example 1. Let C^{PwR} be a collection (as defined in Definition 3),

$$C^{PwR} = \{C^{Patient}\{C^{Patient\_records}\}\}$$

where

$$C^{Patient} = \{(Att, Val)^{Patient}_{Patient\_Id}, (Att, Val)^{Patient}_{NSS\_number}, (Att, Val)^{Patient}_{gender}, \dots\},$$
$$C^{Patient\_records} = \{(Att, Val)^{Patient\_records}_{Patient\_Id}, (Att, Val)^{Patient\_records}_{Patient\_record\_Id}, \dots\}.$$

The detail of the document $D_{Patient\_record}$ is illustrated in Figure 9. For example $D_{Patient\_record}$ provides the number of a patient's record, of the patient named "Younes GUELL" for the date 15/01/2021, having a medical condition equal to *COVID-19*, updated by a staff member identified by his id.

The collection C^{PwR} will be defined as follows:

$$C^{PwR} = \left\{ \begin{array}{c} (Att, Val)^{Patient}_{Patient\_Id}, (Att, Val)^{Patient}_{NSS\_number}, (Att, Val)^{Patient}_{gender}, \\ (Att, Val)^{Patient}_{Full\_name}, \dots, (Att, Val)^{Patient\_records}_{Patient\_Id}, \\ N^{Patient\_records} : \{(Att, Val)^{Patient\_records}_{Patient\_record\_Id}, (Att, Val)^{Patient\_records}_{record\_detail}, \dots\} \end{array} \right\}$$

where $Att^{Patient}_{Patient\_Id}$, $Att^{Patient}_{Full\_name}$, $Att^{Patient}_{gender}$ and $Att^{Patient}_{NSS\_number}$ are simple attributes, and the other attributes are compound attributes. Thus, the values in the nested documents for this example are all atomic values.

$$\begin{aligned} Val^{Patient}_{Patient\_Id} &= P012021L001, \\ Val^{Patient\_records}_{Patient\_record\_Id} &= 2021103, \\ Val^{Patient\_records}_{staff\_id\_Updated} &= DH001K02, \\ Val^{Patient\_records}_{approval\_datetime} &= 15/01/2021, \\ Val^{Patient\_records}_{medical\_condition} &= \textit{COVID-19}, \\ Val^{Patient}_{Full\_name} &= \textit{Younes Guel}. \end{aligned}$$

6 CONCLUSION AND FUTURE WORK

This paper presents a modelling approach based on UML use case and class diagrams dedicated to promote better strategies for data manipulation in NoSQL document databases, including the four Big Data V's.

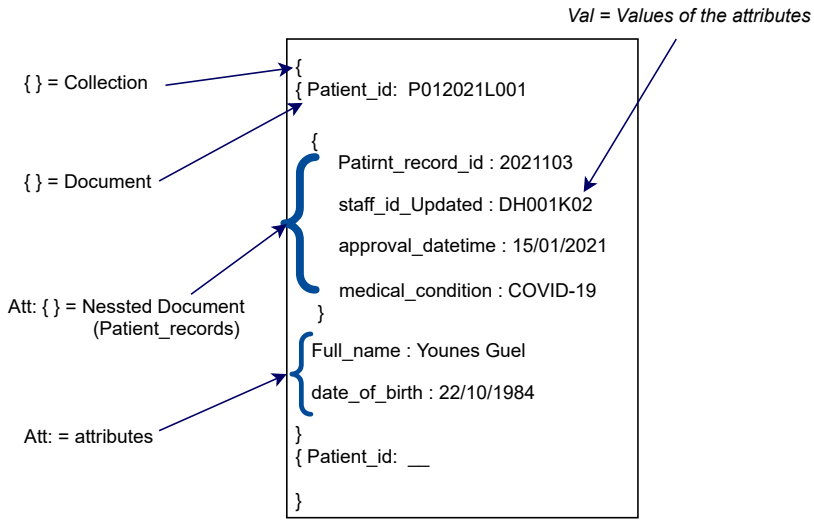

Figure 9. Graphic representation of the collection C^{PwR}

The proposed approach is based on four pillars. We perform a data-up technique resulting in a use case diagram to visualize the behaviour of the system. We also present a conceptual model as an extended class diagram accompanied by guidelines for a successful switching from class diagram to NoSQL document-oriented model. To achieve such result, we introduce some definitions of several terms used to build the document-oriented logical model. To demonstrate the utility of our methodology, we provide an illustrative example.

Unlike related work, the proposed models can readily cover all conceptual details for NoSQL document databases. With future developments, our method will incorporate other types of NoSQL databases. In this paper we have concentrated our attention only on the understanding of the data at hand and the user's needs phases. On the basis of the promising findings, further work on the remaining UML diagrams is underway and will be presented in future works. Also we plan to work on a standard benchmark for the evaluation of our proposal, as well as any future proposals, so that the performance of NoSQL database design becomes measurable and comparable.

REFERENCES

[1] Chen, M.—Mao, S.—Liu, Y.: Big Data: A Survey. Mobile Networks and Applications, Vol. 19, 2014, No. 2, pp. 171–209, doi: 10.1007/s11036-013-0489-0.

[2] Gandomi, A.—Haider, M.: Beyond the Hype: Big Data Concepts, Methods, and Analytics. International Journal of Information Management, Vol. 35, 2015, No. 2, pp. 137–144, doi: 10.1016/j.ijinfomgt.2014.10.007.

[3] George, G.—Osinga, E. C.—Lavie, D.—Scott, B. A.: Big Data and Data Science Methods for Management Research. Academy of Management Journal, Vol. 59, 2016, No. 5, pp. 1493–1507, doi: 10.5465/amj.2016.4005.

[4] Li, S.—Dragicevic, S.—Castro, F. A.—Sester, M.—Winter, S.—Coltekin, A. et al.: Geospatial Big Data Handling Theory and Methods: A Review and Research Challenges. ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 115, 2016, pp. 119–133, doi: 10.1016/j.isprsjprs.2015.10.012.

[5] Storey, V. C.—Song, I. Y.: Big Data Technologies and Management: What Conceptual Modeling Can Do. Data and Knowledge Engineering, Vol. 108, 2017, pp. 50–67, doi: 10.1016/j.datak.2017.01.001.

[6] Rats, J.: Developing and Evaluating ECM Data Persistence Architecture. Computing and Informatics, Vol. 38, 2019, No. 2, pp. 454–472, doi: 10.31577/cai_2019_2_454.

[7] Davoudian, A.—Chen, L.—Liu, M.: A Survey on NoSQL Stores. ACM Computing Surveys (CSUR), Vol. 51, 2019, No. 2, Art. No. 40, doi: 10.1145/3158661.

[8] Shin, K.—Hwang, C.—Jung, H.: NoSQL Database Design Using UML Conceptual Data Model Based on Peter Chen's Framework. International Journal of Applied Engineering Research, Vol. 12, 2017, No. 5, pp. 632–636.

[9] Mior, M. J.—Salem, K.—Aboulnaga, A.—Liu, R.: NoSE: Schema Design for NoSQL Applications. IEEE Transactions on Knowledge and Data Engineering, Vol. 29, 2017, No. 10, pp. 2275–2289, doi: 10.1109/TKDE.2017.2722412.

[10] Chang, F.—Dean, J.—Ghemawat, S.—Hsieh, W. C.—Wallach, D. A.—Burrows, M.—Chandra, T.—Fikes, A.—Gruber, R. E.: Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS), Vol. 26, 2008, No. 2, Art. No. 1000121, doi: 10.1145/1365815.1365816.

[11] Lakshman, A.—Malik, P.: Cassandra: A Decentralized Structured Storage System. ACM SIGOPS Operating Systems Review, Vol. 44, 2010, No. 2, pp. 35–40, doi: 10.1145/1773912.1773922.

[12] Gudivada, V. N.—Rao, D.—Raghavan, V. V.: NoSQL Systems for Big Data Management. 2014 IEEE World Congress on Services, IEEE, 2014, pp. 190–197, doi: 10.1109/SERVICES.2014.42.

[13] Matallah, H.—Belalem, G.—Bouamrane, K.: Evaluation of NoSQL Databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB. International Journal of Software Science and Computational Intelligence (IJSSCI), Vol. 12, 2020, No. 4, pp. 71–91, doi: 10.4018/IJSSCI.2020100105.

[14] Das, A.—Mitra, A.—Bhagat, S. N.—Paul, S.: Issues and Concepts of Graph Database and a Comparative Analysis on List of Graph Database Tools. 2020 International Conference on Computer Communication and Informatics (ICCCI), IEEE, 2020, pp. 1–6, doi: 10.1109/ICCCI48352.2020.9104202.

[15] Roques, P.: UML 2.5 Par La Pratique: Etudes De Cas Et Exercices Corrigés. Editions Eyrolles, 2018 (in French).

[16] Marbán, Ó.—Segovia, J.: Extending UML for Modeling Data Mining Projects (DM-UML). Journal of Information Technology and Software Engineering, Vol. 3, 2013, No. 2, Art. No. 1000121, doi: 10.4172/2165-7866.1000121.

[17] Atzeni, P.—Bugiotti, F.—Cabibbo, L.—Torlone, R.: Data Modeling in the NoSQL World. Computer Standards and Interfaces, Vol. 67, 2020, Art. No. 103149, doi: 10.1016/j.csi.2016.10.003.

[18] McConnell, C. C.—Liu, W.—Shayandeh, S.—Goodwin, R. L.: Efficient Denormalization of Data Instances. Google Patents, 2020 (US Patent App. 16/740,081).

[19] Chevalier, M.—El Malki, M.—Kopliku, A.—Teste, O.—Tournier, R.: Implementing Multidimensional Data Warehouses into NoSQL. Proceedings of the 17th International Conference on Enterprise Information Systems – Volume 2 (ICEIS), 2015, pp. 172–183, doi: 10.5220/0005379801720183.

[20] Dehdouh, K.—Boussaid, O.—Bentayeb, F.: Big Data Warehouse: Building Columnar NoSQL OLAP Cubes. International Journal of Decision Support System Technology (IJDSST), Vol. 12, 2020, No. 1, pp. 1–24, doi: 10.4018/IJDSST.2020010101.

[21] Yangui, R.—Nabli, A.—Gargouri, F.: Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study. Procedia Computer Science, Vol. 96, 2016, pp. 255–264, doi: 10.1016/j.procs.2016.08.138.

[22] Feng, W.—Gu, P.—Zhang, C.—Zhou, K.: Transforming UML Class Diagram into Cassandra Data Model with Annotations. 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), IEEE, 2015, pp. 798–805, doi: 10.1109/SmartCity.2015.165.

[23] Gómez, P.—Casallas, R.—Roncancio, C.: Automatic Schema Generation for Document-Oriented Systems. In: Hartmann, S., Küng, J., Kotsis, G., Tjoa, A. M., Khalil, I. (Eds.): Database and Expert Systems Applications (DEXA 2020). Springer, Cham, Lecture Notes in Computer Science, Vol. 12391, 2020, pp. 152–163, doi: 10.1007/978-3-030-59003-1_10.

[24] Abdelhedi, F.—Brahim, A. A.—Atigui, F.—Zurfluh, G.: MDA-Based Approach for NoSQL Databases Modelling. In: Bellatreche, L., Chakravarthy, S. (Eds.): Big Data Analytics and Knowledge Discovery (DaWaK 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10440, 2017, pp. 88–102, doi: 10.1007/978-3-319-64283-3_7.

[25] Daniel, G.—Sunyé, G.—Cabot, J.: UMLtoGraphDB: Mapping Conceptual Schemas to Graph Databases. In: Comyn-Wattiau, I., Tanaka, K., Song, I. Y., Yamamoto, S., Saeki, M. (Eds.): Conceptual Modeling (ER 2016). Springer, Cham, Lecture Notes in Computer Science, Vol. 9974, 2016, pp. 430–444, doi: 10.1007/978-3-319-46397-1_33.

[26] Krommyda, M. K.—Kantere, V.: The Big Data Era: Data Management Novelties for Visualizing, Exploring, and Processing Big Data. Analyzing Future Applications of AI, Sensors, and Robotics in Society, IGI Global, 2020, pp. 87–103, doi: 10.4018/978-1-7998-3499-1.ch006.

[27] Sharma, A.—Singh, G.—Rehman, S.: A Review of Big Data Challenges and Preserving Privacy in Big Data. In: Kolhe, M., Tiwari, S., Trivedi, M., Mishra, K.

(Eds.): Advances in Data and Information Sciences. Springer, Singapore, Lecture Notes in Networks and Systems, Vol. 94, 2020, pp. 57–65, doi: 10.1007/978-981-15-0694-9_7.

[28] Silva, C. V.—Saens, R.—Del Río, C.—Villarroel, R.: Aspect-Oriented Modeling: Applying Aspect-Oriented UML Use Cases and Extending Aspect-Z. Computing and Informatics, Vol. 32, 2013, No. 3, pp. 573–593.

[29] Stokes, J.: Managing the Development of Large Software Systems – Apollo Real-Time Control Center. 1970.

[30] Yoo, J.—Lee, K. H.—Jeon, Y. H.: Migration from RDBMS to NoSQL Using Column-Level Denormalization and Atomic Aggregates. Journal of Information Science and Engineering, Vol. 34, 2018, No. 1, pp. 243–259, doi: 10.6688/JISE.2018.34.1.15.

[31] Roy-Hubara, N.—Sturm, A.: Design Methods for the New Database Era: A Systematic Literature Review. Software and Systems Modeling, Vol. 19, 2020, No. 2, pp. 297–312, doi: 10.1007/s10270-019-00739-8.

Mohammed ElHabib Maicha is Assistant Professor with the Department of Computer Science and a member of the LIM laboratory, Amar Telidji University Laghouat, Algeria. He received his engineering degree from the University of Laghouat, in 2011. He has been currently pursuing his Ph.D. degree at the University of Laghouat, Algeria, since 2017. His research interests include databases, modeling, Big Data, and NoSQL.

Youcef Ouinten received his M.Sc. degree and Ph.D. degree in operational research from the University of Southampton, UK, in 1984 and 1988, respectively. He received his graduation degree (DES) in mathematics, option operational research, from the University of Science and Technology – Houari Boumediene of Algiers, Algeria, in 1981. He served as Head of the Computing Center at the University Amar Telidji of Laghouat, from 1999 to 2012. He is currently Senior Lecturer at the Department of Mathematics and Computer Science of the University Amar Telidji of Laghouat, Algeria. His research interests include data mining, text mining, information retrieval and optimization.

Benameur Ziani received his Engineer degree in computer science from the Sidi Belabbes University, Algeria, and Ph.D. degree in computer science from the University of Laghouat, Algeria. He is currently Associate Professor in computer science at the Department of Computer Science of the University of Laghouat. Prior to joining the Department of Computer Science he served as Engineer in computer science at the computing center of the University of Laghouat during 1992–2012. His current research interests include knowledge discovery, data mining and machine learning with applications in various areas: database and data-warehouse design optimisation, Big Data and data networks analytics.