International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-10 Issue-3, February 2021

The Integration of the MDA Approach in Document-Oriented NoSQL Databases, the case of Mongo DB

Aziz Srai, Fatima Guerouate, Hilal Drissi Lahsini

Abstract*: Today with the growth of the internet, the use of social networks, mobile telephony, connected and communicating objects. The data has become so big, hence the need to exploit that data has become primordial. In practice, a very large number of companies specializing in the health sector, the banking and financial sector, insurance, manufacturing industry, etc… are based on traditional databases which are often well organized of customer data, machine data, etc ... but in most cases, very large volumes of data from these databases, and the speed with which they must be analyzed to meet the business needs of the company are real challenges. This article aims to respond to a problem of generating NoSQL MongoDB databases by applying an approach based on model-driven engineering (Model Driven Architecture Approach). We provide Model to Model (using the QVT model transformation language), and Model to Code transformations (using the code generator, Acceleo). We also propose vertical and horizontal transformations to demonstrate the validity of our approach on NoSQL MongoDB databases. We have studied in this article the PSM transformations towards the implementation. PIM to PSM transformations are the subject of another work.

***Keywords* : *MDA, NoSQL,* Document-oriented Databases, MongoDB.**

I. INTRODUCTION

Since their creation, databases, small or large, have become an essential entity and inseparable from any application or website. The basics of most popular relational data at the time had their DBMSs available by default in computer systems. With the expansion of the number of Internet users and the multitude of terminals and connected objects, relational databases are no longer able to support large data to store, extract, move and copy, especially if they are distributed over multiple servers. Hence the need for a new generation of advanced databases, compatible with the geographic extent of huge networks of servers, known as clusters, and capable of handling large amounts of data. Fig.1 shows the evolution of information systems and data exchanged from the creation of computer networks to the new generation of the Web, allowing Internet users to contribute to the exchange of information and to interact in a simple way, both in terms of content and structure of the pages, in particular creating the social Web.

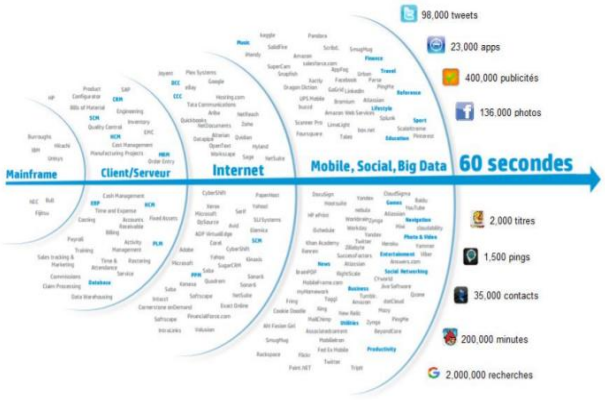

Fig. 1. Exponential expansion of data exchanged on the Internet.

Today, the ubiquity of the Internet connection is a reality (the cars we drive, the watches we wear, our small household medical devices, our refrigerators and freezers, our smartphones and laptops). In addition, digital data produced by humans, including video footage, photos and more, attains large volumes per day. These data currently stored in databases that have been designed specifically for them are managed by large database management software, playing the role of intermediaries between the databases on the one hand and the applications and their users on the one hand. Other, we are talking here about non-relational databases, called NoSQL. In this paper we propose the integration of an MDA approach in the context of NoSQL databases in particular the case of MongoDB document-oriented databases. This paper is organized as follows: we begin in the first section with an introduction. The section 2 discusses the works that are related to our theme. Section 3 presents the concepts of the MDA approach (Model Driven Approach). Sections 4 and 5 present our proposed solution to develop E-learning platform. The final section concludes this paper, and outlines future work.

II. RELATED WORKS

Previously, several research projects have been proposed in the context of integrating the MDA approach into NoSQL databases.

Manuscript received on February 09, 2021.
Revised Manuscript received on February 18, 2021.
Manuscript published on February 28, 2021.
* Correspondence Author
Dr. Aziz Srai*, Research Doctor at LASTIMI Laboratory, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco.: aziz.srai.dev@gmail.com
Prof. Fatima Guerouate, Research Professor at LASTIMI Laboratory, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco. Email: guerouate@gmail.com
Prof. Hilal Drissi Lahsini, Research Professor at LASTIMI Laboratory, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco. Email: Hilaldrissi@gmail.com

In [Chevalier, 2015], the authors defined a set of rules for mapping a star schema into two NoSQL models: column-oriented and document-oriented. Other studies [Li et al., 2010] and [Vajk et al., 2013] have studied the process of transforming relational databases into a NoSQL model. [Li et al, 2010] proposed an approach to transform a relational database into HBase (column oriented system). [Vajk et al., 2013] defined a correspondence between a relational model and a document-oriented model using MongoDB. [Li et al., 2014] propose a MDA-based process to transform UML class diagram into column-oriented model specific to HBase. [Gwendal et al., 2016] describe the mapping between a conceptual UML model and graphical databases via an intermediate graphical metamodel. In this work, transformation rules are specific to graphical databases used as a framework for managing complex data with many connections. Generally, this type of NoSQL system is used in social networks where data is highly connected. According to our knowledge, no work has studied the transformation of a uml diagram to a NoSQL MongoDB database, nor the generation of a NoSQL MongoDB database through a PSM transformation to an implementation.

III. MODEL DRIVEN ENGINEERING

Since the end of the 90s, Model Driven Engineering (MDE) has been regarded as an undisputed approach to dealing with the complexity of distributed systems and this is based on two very important principles in software development which are: abstraction and automation. The abstraction is based on the representation of systems as a model to facilitate understanding of the architecture and behavior of these systems. This approach is completely model-based. These models will serve as a starting point in the process of specification, development and analysis. They can be used to understand, evaluate, communicate, and produce code. These automated transformations increase productivity and decrease development costs. The MDE emphasizes domain-specific models, which may be more useful for specifying applications and generating code. In order to model complex systems of reasonable size, designers need to separate the model into multiple views, each of which captures a specific system concern. These different views of the model are governed by viewpoints and are used to facilitate the tasks of designing, analyzing and developing software.

Model Driven Engineering (MDE) is a branch of software engineering concerned with the development, maintenance and evolution of models within computer systems.

The first software architecture following these principles is called MDA (Model Driven Architecture). It was proposed by the Object Management Group (OMG), to succeed the Object Management Architecture. This architecture, proposed in 2000, is the variation of Model Engineering within the OMG. We could summarize the approaches of MDA and Model-driven engineering as follows:

1. The concept of model;
2. The concept of metamodel;
3. The concept of model transformation.

The "model" is a simplified representation of a studied system. The system being the entity that is modeled in order to study, understand and use it for predictive purposes in a controlled context other than the real context. [Minsky 1965] proposed the following definition: "For an observer B, an object A * is a model of an object A, since B can use A * to answer questions of interest to him about A". [Popper 1973] thus summarizes three characteristics common to all models:

1. A model must have a character of resemblance to the real system;
2. A model should be a simplification of the real system;
3. A model is an idealization of the real system.

Particularly in the context of the model-driven engineering, these three characteristics can be expressed by the relationship “is a representation of” between the model and the system studied ([Hill 1996] [Atkinson and Kuhne, 2003] [Seidewitz 2003] [Bézivin 2004]). A model is not assumed be perfect or contain all the information about the system being studied. It is simply considered "good enough" for a certain purpose [Favre 2004].

However, this definition is not sufficient within the framework of the model-driven engineering because it does not make it possible to make a model "productive". This is why some authors use the following definition [Kleppe et al., 2003]:

"A model is a description of a (or part of a) system written in a well-defined language".

The notion of well-defined language indirectly appeals to the second principle of the model-driven engineering, namely the notion of metamodel.

As for the concept of model, several definitions exist in the literature:

“A metamodel is a model that defines a language to express a model” [OMG 2002].

“A metamodel is a specification model for a class of studied systems, where each studied system in that class is itself a valid model expressed in a certain modeling language” [Kleppe et al., 2003].

Contrary to popular belief, the metamodel in this context is not a model of models, as it might be when we were talking about statistical methods. Rather, we could define the metamodel as the model of a modeling language. This definition is based on the following typical relation, of the notion of metamodel: "a model conforms to a metamodel".

In an identical way to what we have seen for the notion of metamodel, the transformation takes different forms depending on the technical spaces or levels of abstraction studied:

- For databases, we find the migration scripts.
- For the XML language: the transformations are expressed in eXtensible Stylesheet Language (XSLT).
- For language theory: transformations are handled by compilation and more particularly by code generation.
- For models (M2M, "Model to Model"): there are languages such as ATLAS Transformation Language (ATL) or QVT (Query View Transform).

Among the most popular meta-modeling languages are the Meta Object Facility, UML, Eclipse Modeling Framework specification. In the case of EMF, for example, transforming models amounts to running a metaprogram. It is, in effect, a modeling and code generation platform.

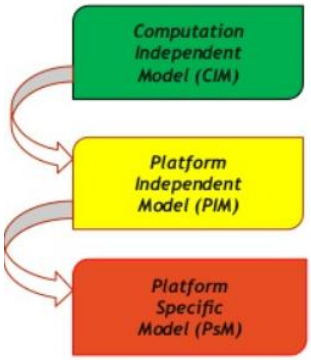

Fig .2 . models of the MDA approach.

IV. BIG DATA

Big Data is a type of data base with specific characteristics. In this section, we present the three rules that have been widely used to define Big Data. These are the "3V", "4V" and "5V" rules.

- "3V" Rule: A description of Big Data was established in a 2001 research report by Gartner Group analyst Douglas Laney [Douglas, 2001]. She defines the issues inherent in data growth as three-dimensional. These are Volume, Variety and Velocity. In 2012, Gartner1 gave a more detailed definition of Big Data as follows:

Definition 1. "Big Data is big data, very varied, generated and processed at high speed. These data require efficient and innovative forms of information processing to enable better decision-making " [Douglas, 2001].

- "4V" rule: The analysis firm IDC (International Data Corporation) defined, in a 2011 report [Gantz and Reinsel, 2011], four dimensions to characterize Big Data, namely: Volume, Variety, Velocity and Value.

Definition 2. "Big Data Technologies describe a new generation of technologies and architectures designed to economically extract value from large volumes of very varied data, allowing their capture and analysis at high speed" [Gantz and Reinsel , 2011].

- "5V" rule: Two new Vs appeared in 2013 in a research article [Demchenko et al., 2013] where the SNE group (System and Network Engineering) proposes a broader definition for Big Data through the rule of 5V.

Definition 3. "Big Data Technologies aim to process large volume, high velocity and wide variety data to extract value, ensure high veracity of original data and obtain information that requires innovative forms of data processing, in order to improve decision-making and process control " [Demchenko et al., 2013].

The definitions associated with the dimensions used in the above rules are as follows:

- Volume: represents the size of all the data to be processed.
- Variety: refers to the variety of sources, types and formats of data.
- Velocity: corresponds to the speed at which data is collected and processed.
- Value: equivalent to the profit that can be derived from the use of Big Data.
- Veracity: covers the quality and reliability of the data; this is the qualitative dimension of Big Data.

Currently 90% of DBMS are relational. But faced with the characteristics of Big Data, relational systems encounter limits. The main problems with these systems are: Horizontal scaling: a relational DB was primarily designed for single server configurations [Kumar et al., 2015]; scaling this database involves distributing it across multiple servers. This poses financial (the cost of servers) and technical constraints (the number of servers is generally limited to 10). In addition, managing tables on different servers remains a complex task. The definition of a model during the creation of the database and before data entry: in a Big Data context, the user must be able to easily integrate new data. Relational DBMSs do not offer this flexibility; the relational model is difficult to incrementally change without affecting performance or taking the DB offline. Thus, storage technologies have had to evolve to introduce new DBMSs which are capable of handling large, varied data and which can evolve very quickly. These are the NoSQL DBMS [Han et al, 2011] presented in the next section.

V. NOSQL DATABASES

The term NoSQL refers to a type of database management system that goes beyond the relational systems associated with the SQL language by accepting more complex data structures. According to their physical models, the databases managed by these systems are divided into four categories: columns, documents, graphs and key-value [Angadi et al., 2013]. Each of them offering specific features. For example, in a document-oriented DB like MongoDB, the data is stored in tables whose rows can be nested. This data organization is coupled with operators that provide access to nested data [Kumar et al., 2015].

The choice of the most suitable DBMS category for a given application is linked to the nature of the processing (requests) applied to the data. But this choice is not exclusive since, in each category, the DBMS can provide all types of processing, sometimes at the cost of a certain cumbersome or more extensive programming.

In what follows, we present the data models adopted by each category of NoSQL DBMS.

A. Column-oriented model

The column-oriented model is a structured model where data is organized into families of columns, which is equivalent to the concept of a table in the relational model. The lines have an identifier called the line key and are made up of a set of values; each is associated with a column. Thus, finding a value amounts to going through the sequence: row key -> family of columns -> column.

Although the column-oriented model is similar to the relational model, the organization of the data in the two models is different [Abadi et al., 2008]. In opposition to what is found in a relational database where the columns are static and present in each row, in a column-oriented database the columns are dynamic and appear only in the rows concerned. In other words, each row has a different number of columns and you can add new columns to it at any time; we thus gain in extensibility at the level of the data model.

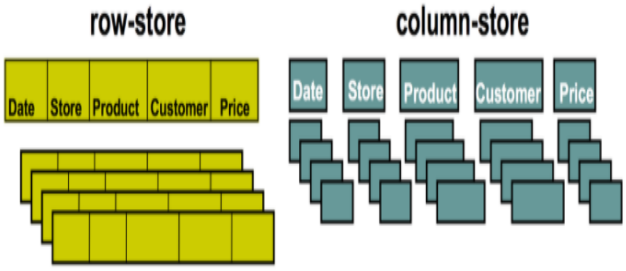

Fig. 3. Organization of a family of columns in a column-oriented database.

In addition, the column-oriented model has the advantage of improving storage efficiency and avoiding space consumption compared to the relational model. Indeed, due to their design by allocation of blocks, in a relational DBMS, an empty column will still consume space. In a column-oriented DBMS, the cost of storing an empty column is 0. Cassandra, HBase and Accumulo are examples of DBMS where data is stored in a column-oriented model.

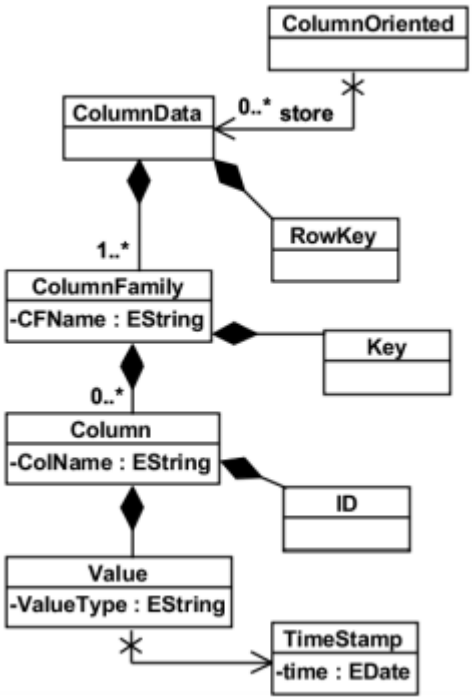

Fig. 4. Meta-model for Column-oriented database [Erraissi et al et al., 2019].

B. Document-oriented database

Document-oriented databases are used to manage semi-structured data. This is data that does not follow a fixed structure and carries the structure within it. However, markings in the semi-structured data make it possible to organize the information. Due to the lack of a clear structure, these data are not suitable for relational databases since their information cannot be organized in tables.

Document-oriented databases create a simple pair: a key is assigned to a specific document. This document, which can for example be formatted with XML, JSON or YAML contains the information itself. Since the database does not require a specific schema, it is also possible to integrate different types of documents in a single document store. Changes to documents do not have to be communicated to the database.

1) Operation of document-oriented databases

In theory, it is possible to place data of different formats and without a consistent schema in a document-oriented database. In practice, a file format is generally used for documents and the information is integrated into a fixed structure. This makes it easier to work with the information and the database. This organization makes it possible, for example, to process database search queries more efficiently. In general, a document-oriented database can perform the same actions as a relational system: information can be inserted, modified, deleted and searched. To be able to perform these actions, each document receives a unique identifier. How it is designed is in principle not important. It is possible to use both a sequence of characters and a full path to address the document. When searching for information, it is the documents themselves that are analyzed: this means that the corresponding data is not looked for in several columns of the database and is extracted directly from the documents.

2) Advantages and disadvantages of document-oriented databases

In classic relational databases, a field must exist for each information and in each entry. When the information is not available, the cell remains empty and must be created. Document-oriented databases are much more flexible: the structure of different documents does not have to be consistent. The database can even host large volumes of unstructured data.

Furthermore, it is very easy to integrate new information: whereas with a relational database, it is necessary to insert a new information point in all the data sets, in the case of a document store, it is sufficient to embed the new entry in only a few datasets. In other documents, additional content can be added, but it is not essential.

Moreover, in the case of document stores, the information is not shared between several tables linked to each other. Everything is in one place which can lead to better performance. However, document-oriented databases can only exploit this speed advantage as long as you don't try to endow them with relational elements: references cannot be used in the concept of document stores. If you try to link documents together, then the system becomes very complex and impractical. In the case of strongly linked data volumes, a relational database is therefore more appropriate.

3) The most famous document-oriented databases

For web application development in particular, databases play a huge role for documents. Due to the high demand resulting from web development, many database management systems (DBMS) are now available in the market. The following list presents the most famous:

- BaseX: this open source project uses Java and XML.
- BaseX comes with a graphical user interface.
- CouchDB: The Apache Software Foundation released the open source software CouchDB. This database management system is coded in Erlang, uses JavaScript and is used in particular in Ubuntu and in Facebook applications.

Retrieval Number: 100.1/ijeat.C22350210321
DOI:10.35940/ijeat.C2235.0210321
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication

- Elasticsearch: This search engine works on the basis of a document-oriented database. To do this, JSON documents are used.
- eXist: The open source eXist database management system runs through a Java virtual machine and can therefore be used independently of an operating system. The documents used are mainly in XML format.
- MongoDB: MongoDB is the most popular NoSQL database. The software is coded in C ++ and uses documents similar to JSON.
- SimpleDB: With SimpleDB (encoded in Erlang), Amazon has developed its own database management system for corporate cloud services. The provider charges a fee for its use.

Fig. 5. Organization of a collection in a document-oriented database.

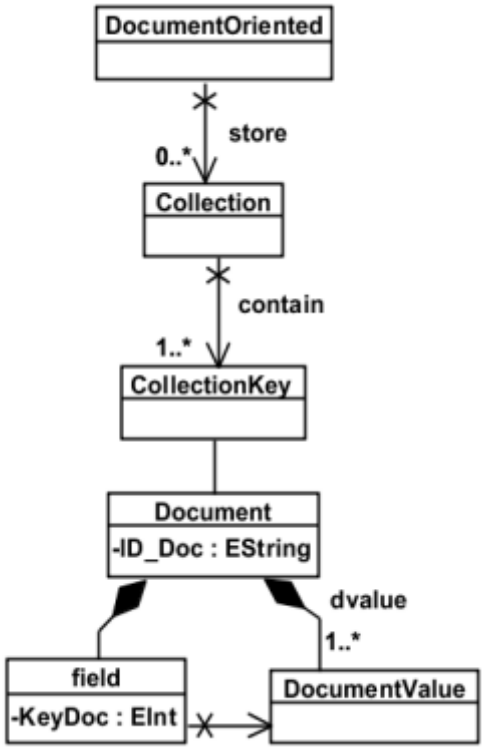

Fig .6. Meta-model for Document-oriented database [Erraissi et al et al., 2019].

C. Graph-oriented model

This model organizes data in the form of nodes and relationships. Nodes and arcs can carry a set of properties expressed as key-value pairs.

This model is useful for storing and querying complex, tightly related data; this is the case, for example, with data from social networks.

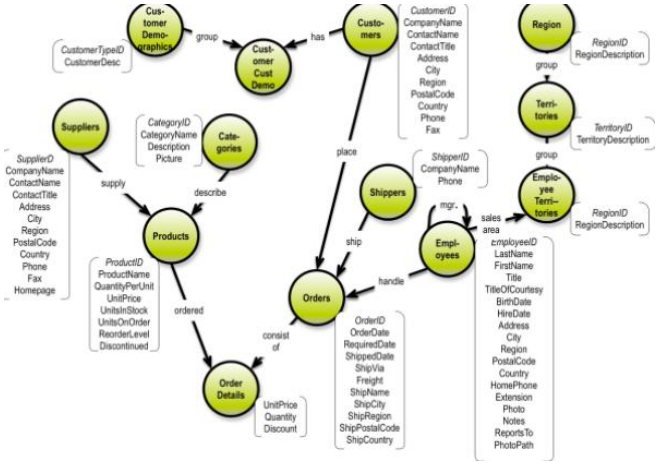

Fig. 7. Organisation des données dans une BD orientée-graphes.

The most common graph-oriented DBMS are Neo4j, OrientDB and FlockDB.

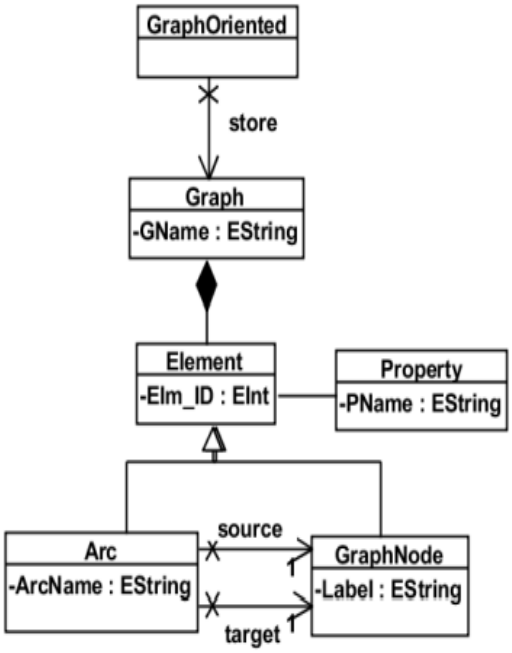

Fig .8. Meta-model for Graph-oriented database [Erraissi et al et al., 2019].

D. Key-value oriented model

This is the most basic NoSQL model and was the forerunner in NoSQL DBMS. It organizes data in the form of key-value pairs, where the key is the single point of entry that accesses the data. The column, document and graph oriented data models presented previously are evolutions of the key-value model. Due to their simplified access model that exclusively uses the key, DBMSs that adopt the key-value model, such as Redis and Riak, achieve high performance in terms of data access time. However, they only offer simplified functionalities in terms of query expression [Angadi et al., 2013]. Thus, these DBMS are mainly used in contexts where the requirements in terms of requests are very low.

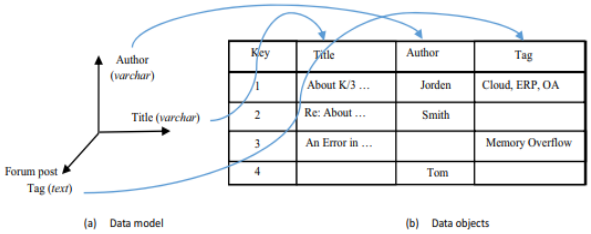

Fig .9. An example of key-value based data model [Bo Hu et al., 2014].

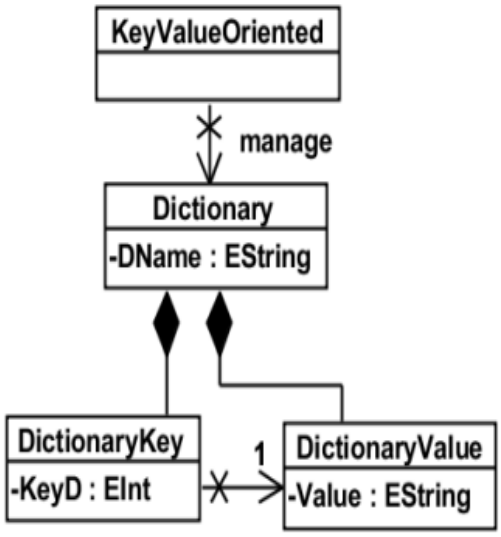

Fig .10. Meta-model for Key/Value database [Erraissi et al et al., 2019].

1) metamodel document-oriented databases

The authors in [Ait Brahim et al., 2019] have proposed an algorithm to represent document-oriented databases; the proposed algorithm is represented below:

Definition 1. A mongodb BD^{md} database is defined by (N, CLL) where:

- N is the name of the base, - CLL = $\{cll_1,\ldots, cll_n\}$ is a set of collections.

Definition 10. A mongodb $BDmd$ database is defined by (N, CLL) where:

- N is the name of the base,
- CLL = $\{cll_1,\ldots, cll_n\}$ is a set of collections.

Definition 2. ∀ i ∈ [1..n], the scheme of a collection $clli$ ∈ CLL is a pair (N, FL) where:

- cll_i .N is the name identifying the collection,
- cll_i .FL = $FL^a \cup FL^{cx} \cup \{Id^{cll}\}$ is the set of atomic fields Fl^a = $\{fl_1{}^a,\ldots, fl_r{}^a\}$ and complexes Fl^{cx} = $\{fl_1{}^{cx},\ldots, fl_s{}^{cx}\}$ which will be used to define the documents of cll_i, where:
- ∀ i ∈ [1..r], the diagram of an atomic field $fl_i{}^a \in FL^a$ is a couple (N, Ty) where:
- $fl_i{}^a$ .N is the name identifying the field,
- $fl_i{}^a$ .Ty is the type of field.
- ∀ j ∈ [1..s], the scheme of a complex field $fl_j{}^{cx} \in FL^{cx}$ is a couple (N, FL ') where:
- $fl_j{}^{cx}$.N is the name identifying the field,
- $fl_j{}^{cx}$.FL 'is the set of nested fields in $fl_j{}^{cx}$ where FL'⊂FL.
- Id^{cll} is a special field which must exist in each document of cll_i in order to uniquely identify it in the collection. This field has a fixed name noted _id.

VI. PROPOSED METHODOLOGY

To generate document-oriented NoSQL databases platform through an MDA approach we have considered the class diagram Fig. 11. We consider the class diagram illustrated in this figure sufficient to apply the MDA approach on document-oriented NoSQL databases, and letting the paper quite understandable and clear.

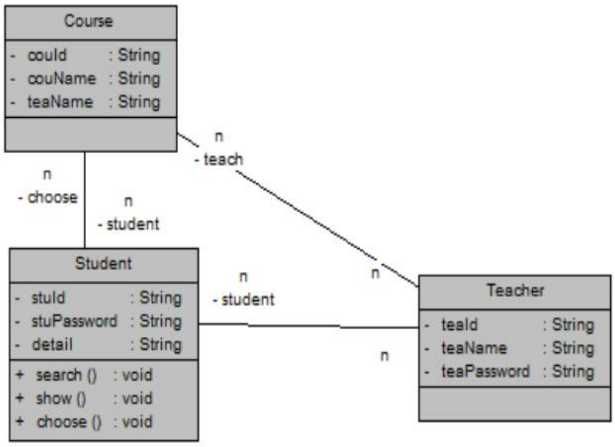

Fig. 11. Class diagram used in the generation of document-oriented NoSQL databases.

Then we have defined the metamodel for document oriented NoSQL platforms illustrated in Fig. 12 :

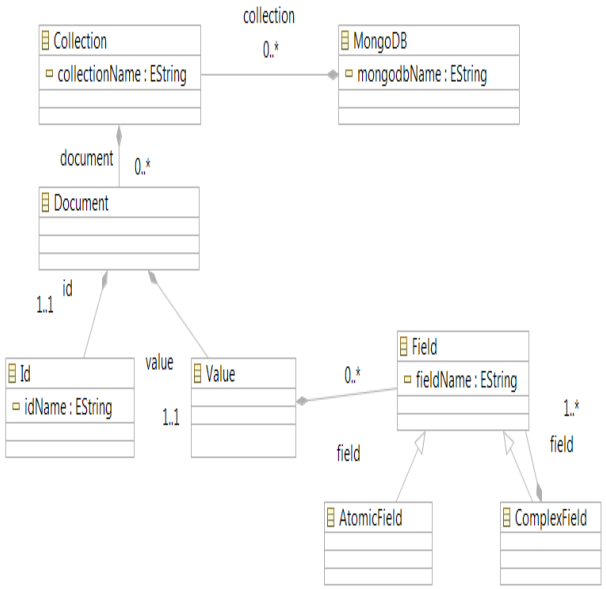

Fig. 12 . MongoDB Metamodel.

We have defined also the different transformation rules from the PSM model to implementation in Acceleo Query Language (AQL) illustrated in Fig. 13:

Retrieval Number: 100.1/ijeat.C22350210321
DOI:10.35940/ijeat.C2235.0210321
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication

```
[comment encoding = UTF-8 /]
[module generate('http://mongodb.mm')]

[template public generateElement(aMongoDB : MongoDB)]
[comment @main/]
[file (aMongoDB.mongodbName.concat('.mjs'), false, 'UTF-8')]
[for (aMongoDB.collection)]
 {
  "[aMongoDB.collection.document.id.idName.toUpperFirst()/]" : "[aMongoDB.collection.document.value.field.fieldName.toUpperFirst()/]",

[comment .../]
 }

[/for]
[/file]
[/template]
```

Fig. 13 . Generation of Mongo document-oriented NoSQL databases with Acceleo.

VII. RESULT AND DISCUSSION

After all the transformation we have done, we have generated the code to implement a NoSQL database complemented with MongoDB document, we are preparing in a future work a transformation which generalizes the model to model transformations (PIM to PSM) with the transformation language QVT. Remember that the context of this work is a model to Text transformation (PSM to implementation).

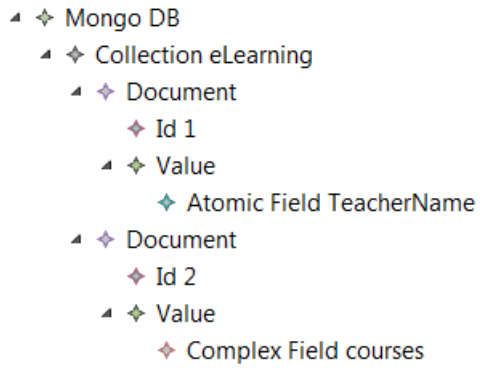

Fig. 13 . PSM model for MongoDB generated from Acceleo.

```
{
"Id": "1",
"last_name": "Srai",
"job":"Teacher",
"age": 33
course : {
     "Title":"Java ",

     }
}
{
"Id": "2",
"last_name": ".....",
"age": 50
course :
     "Title":"MDA ... ",

 }
```

Fig. 14 . Implementation model for MongoDB document-oriented NoSQL database.

VIII. CONCLUSION

In this paper, we have provided an automatic approach that guides and eases the task of implementing a MongoDB document-oriented NoSQL database. This approach is based on MDA notably known as a framework for automatic model transformations. Our approach provides a set of model-to-implementation (M2T) transformations using Acceleo as a code generator. As part of future work, we plan to complete our transformation process in order to take into account the different NoSQL databases such as column-oriented databases (Cassandra), Graph-Oriented Bases (Neo4j), and key-value data (DynamoDB).

REFERENCES

1. Kim, S.D., Min, H.G., Her, J.S., Chang, S.H.: Dream : a practical product line engineering using model driven architecture. In: Proceedings of the Third International Conference on Information Technology and Applications (ICITA 2005) (2005).
2. A. Srai et al, 2017, " MDA approach for CodeIgniter PHP Framework ", The 2 nd Scientific Day on Computer Science, Optimization and Systems' Modelization (CSOSM'17), March 9, 2017- Faculty of science, Kenitra.
3. J. A. Monte-Mor, E. O. Ferreira, H. F. Campos, A. M. da Cunha, and L. A. V. Dias, "Applying MDA Approach to Create Graphical User Interfaces", Eighth International Conference on Information Technology: New Generations, Las Vegas, NV, IEEE, (2011) April 11-13 , p.p. 766-771.
4. X. Blanc, MDA en action : Ingénierie logicielle guidée par les modèles. 1st edition, 270 pages, 2005.
5. Czarnecki, K., Helsen, S., Classification of Model Transformation Approaches, in online proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA. Anaheim, October, 2003.
6. Gharavi, V., Mesbah, A., Deursen, A. V., Modelling and Generating AJAX Applications: A Model-Driven Approach. Proceeding of the7th International Workshop on Web-Oriented Software Technologies, New York, USA (Page: 38, Year of publication: 2008, ISBN: 978-80-227-2899-7).
7. Cong, X., Zhang, H., Zhou, D., Lu, P., Qin, L., A Model-Driven Architecture Approach for Developing E-Learning Platform , Entertainment for Education. Digital Techniques and Systems Lecture Notes in Computer Science, Volume 6249/2010, 111-122, DOI: 10.1007/978-3-642-14533-9_12, 2010.
8. Frédéric J., & Ivan, K. (2006). Transforming models with ATL. Proceedings of MoDELS 2005 Workshops, LNCS 3844, (pp. 128 – 138), Springer-Verlag Berlin Heidelberg.
9. Liliana Favre, Liliana Martinez, Claudia Pereira, "Modernizing software in science and engineering: From C/C++ applications to mobile platforms," Conference: ECCOMAS Congress 2016 European Congress on Computational Methods in Applied Sciences and Engineering, DOI: 10.7712/100016.2402.4906.
10. EMF, Eclipse Modeling Framework, viewed September 2014.http://eclipse.org/modeling/emf/.
11. Acceleo, viewed September 2014. https://eclipse.org/acceleo/.
12. Object Management Group (OMG). Model Driven Architecture. Retrieved December 12, 2013, from http://www.omg.org/mda/.
13. [Bézivin, 2004], "Sur les principes de base de l'ingénierie des modèles", DOI: 10.3166/objet.10.4.145-157.
14. [Kleppe et al., 2003], "MDA Explained: The Model Driven Architecture: Practice and Promise". Addison-Wesley Longman Publishing Co., Inc.75 Arlington Street, Suite 300 Boston, MAUnited States, ISBN:978-0-321-19442-8.
15. Daniel, G., Sunyé, G., Cabot, J.: UML to GraphDB: Mapping conceptual schemas to graph databases. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 430–444. Springer, Cham (2016). doi:10.1007/978-3-319- 46397-1_33.
16. Li, Y., Gu, P., Zhang, C.: Transforming UML class diagrams into HBase based on meta-model. In: ISEEE (2014).
17. Vajk, T., Feher, P., Fekete, K., Charaf, H.: Denormalizing data into schema-free databases. In: CogInfoCom (2013).
18. Li, C.: Transforming relational database into HBase: A case study. In: ICSESS (2010).

19. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, R.: Implementing multidimensional data warehouses into NoSQL. In: ICEIS (2015).
20. Ait Brahim, Amal and Tighilt Ferhat, Rabah and Zurfluh, Gilles MDA process to extract the data model from a document-oriented NoSQL database. (2019) In: ICEIS 2019: 21st International Conference on Enterprise Information Systems, 3 May 2019 - 5 May 2019 (Heraklion, Greece).
21. [Abdelhedi et al et al., 2017] ,MDA-Based Approach for NoSQL Databases Modelling, Springer International Publishing AG 2017 L. Bellatreche and S. Chakravarthy (Eds.): DaWaK 2017, LNCS 10440, pp. 88–102, 2017. DOI: 10.1007/978-3-319-64283-3_7.
22. Allae Erraissi et al., International Journal of Advanced Trends in Computer Science and Engineering, 8(3), May - June 2019, 646 – 653.
23. D. M. Blei, A. Y. Ng, M. I. Jordan, "Latent Dirichlet allocation," Journal of Machine Learning Research, 2003, 3: 993-1022.
24. B. Evelson and N. Norman, "Topic overview: business intelligence," Forrester Research, 2008.
25. P. Zikopoulos and C. Eaton, "Understanding big data: Analytics for enterprise class hadoop and streaming data," McGraw-Hill Osborne Media, 2011.
26. [Bo Hu et al., 2014], A Key-Value based Application Platform for Enterprise Big Data, 2014 IEEE DOI 10.1109/BigData.Congress.2014.71.
27. Abadi, D. J., Madden, S. R., & Hachem, N. (2008). Column-stores vs. row-stores: How different are they really? In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 967-980). ACM.
28. Angadi, A. B., Angadi, A. B., & Gull, K. C. (2013). Growth of New Databases & Analysis of NOSQL Datastores. International Journal of Advanced Research in Computer Science and Software Engineering, 3, 1307-1319.
29. Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In Pervasive computing and applications (ICPCA), 2011 6th international conference on (pp. 363-366). IEEE.
30. Kumar, R., Charu, S., & Bansal, S. (2015). Effective way to handling big data problems using NoSQL Database (MongoDB). Journal of Advanced Database Management & Systems, 2(2), 42-48.
31. Demchenko, Y., Ngo, C., & Membrey, P. (2013). Architecture framework and components for the big data ecosystem. Journal of System and Network Engineering, 1-31.
32. Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. IDC iview, 1142(2011), 1-12.
33. Douglas, L., 2001. 3d data management: Controlling data volume, velocity and variety. Gartner. Retrieved, 6, 2001.

AUTHORS PROFILE

Dr. Aziz Srai*, is a research Doctor at LASTIMI Laboratory, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco.: aziz.srai.dev@gmail.com

Prof. Fatima Guerouate, is a Research Professor at LASTIMI Laboratory, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco. Email: guerouate@gmail.com

Prof. Hilal Drissi Lahsini, is a Research Professor at LASTIMI Laboratory, Mohammadia School of engineering, Mohamed V University city of Rabat, Morocco. Email: Hilaldrissi@gmail.com