Extraction process of conceptual model from a document-oriented NoSQL database

Amal Aït Brahim, Rabah Tighilt Ferhat, Gilles Zurfluh

Extraction process of conceptual model from a document-oriented NoSQL database

Amal AIT BRAHIM
Toulouse Institute of Computer Science Research (IRIT)
Toulouse Capitole University, Toulouse, France
amal.aitbrahim@ut-capitole.fr

Rabah TIGHILT FERHAT
Toulouse Institute of Computer Science Research (IRIT)
Toulouse Capitole University, Toulouse, France
rabah.tighilt-ferhat@ut-capitole.fr

Gilles ZURFLUH
Toulouse Institute of Computer Science Research (IRIT)
Toulouse Capitole University, Toulouse, France
Gilles.Zurfluh@ut-capitole.fr

***Abstract* - NoSQL systems are used to manage massive databases that verify 3V: Volume, Variety and Velocity. Generally, these systems are known by the characteristic "schema less" which means that we can create a database without defining the data schema beforehand. This property offers more flexibility and speed by allowing the evolution of the data model during the exploitation of the base. However, to formulate queries on the database, the user needs a precise knowledge of data model. In this article, we propose a process for the automatic extraction of the conceptual model of a document-oriented NoSQL database. To do this, we use the Model Driven Architecture (MDA) architecture that provides a formal framework for automatic model transformation. From a NoSQL database, we propose a set of transformation rules with QVT to generate the conceptual model in the form of a UML class diagram. An experimentation of the extraction process was carried out on an application in the medical field..**

Keywords - Big Data, NoSQL, model extraction, schema less, MDA, QVT.

I. INTRODUCTION

Big Data have attracted a great deal of attention in recent years thanks to the huge amount of data managed, the types of data supported and the speed at which this data is collected and analyzed. This has definitely impacted the tools required to store Big Data, and new kinds of data management tools i.e. NoSQL systems have arisen [5]. Compared to existing DBMSs, NoSQL systems are generally accepted to support greater data volume and to ensure faster data access, undeniable flexibility and scalability [1].

One of the NoSQL key features is that databases can be schema-less. This means, in a table, meanwhile the row is inserted, the attributes names and types are specified. Unlike relational systems - where first, the user defines the schema and creates the tables, second he inserts data -, the schema-less property brings a flexibility that facilitates the physical schema evolution. End-users are able to add information without the need of database administrator. For instance, in the medical program that follows-up patients suffering from a chronic pathology – case of study detailed in Section 2 – one of the benefits of using NoSQL databases is that the evolution of the data (and schema) is fluent. In order to follow the evolution of the pathology, information is entered regularly for a cohort of patients. But the situation of a patient can evolve rapidly which needs the recording of new information. Thus, few months later, each patient will have his own information, and that's how data will evolve over time. Therefore, the data model (1) differs from one patient to another and (2) evolves in unpredictable way over time. We should highlight that this flexibility concerns the physical level i.e. the stored database exclusively [8].

In information systems, the importance and necessity of conceptual models are widely recognized. The conceptual model is a representation at a higher level of abstraction and a semantic knowledge element, which ensures efficient data management [3]. Furthermore, this model is a document of interchange between end-users and designers, and between designers and developers. Also, the conceptual model is used for system maintenance and evolution that can affect business needs and/or deployment platform. The Unified Modeling Language (UML) is widely accepted as the standard of information system modeling.

On the one hand, NoSQL systems have proven their efficiency in managing Big Data. On the other hand, the needs of a conceptual modeling and design approach remain up-to-date. Therefore, we are convinced that it's important to provide a precise and automatic approach that guides and facilitates the reverse engineering task within NoSQL systems. Indeed, in a Big Data context, the evolution of the needs of the user evolve in an exponential way. This evolution requires an update of the data models (adding, modifying or deleting an element of the model); It can be at two levels: conceptual and physical. According to the principles of MDA, if the needs evolve at the conceptual level, it is enough to restart the transformation process to arrive at the new model of implantation corresponding. When it comes to an evolution of the physical model, it is necessary to consider a process of retro-design to adapt the conceptual model to the modifications made on the physical model. This is useful for a user to formulate their data access requests based on an updated conceptual model.

To answer this problem, we propose the "ToConceptualModel" MDA-based approach that starts from a NoSQL physical model (PSM) and extracts the NoSQL

database conceptual model (PIM) using a set of QVT (Query-View-Transformation) rules. As discussed in section 3 (related work), few solutions have dealt with the NoSQL databases conceptual modeling. To the best of our knowledge, none of the proposed solutions has treated links between several tables. In this paper, we present a solution that deals with two types of links (association and composition) between tables stored on MongoDB system.

The rest of the paper is structured as follows. Section 2 presents the case study in the medical field that motivated our work. Section 3 reviews previous work. Section 4 presents our reverse engineering process. Finally, Section 5 concludes the paper and announces future work.

II. ILLUSTRATIVE EXAMPLE

We resume a case study already presented in a previous paper [2] to illustrate our work. It is a medical application that manages international scientific programs to monitor patients with serious diseases. These programs consist of collecting data from hospitals on disease trends, studying the interactions between them and evaluating the results of their treatment over time. These programs collect data that verifies 3V of Big Data. In fact, the institutions involved in these programs generate a quantity of data that can reach several terabytes. In addition, these data are of different types: structured such as respiratory rate, blood pressure of patients, semi-structured such as drug leaflets and unstructured such as referral records, paper prescriptions and radiology reports. Finallemnet, some data is generated continuously by sensors; which means that they are generally sensitive to the time factor. So you have to treat them almost in real time to get the best results.

This case study is a typical application in which the use of a NoSQL system is appropriate. On the one hand, the database contains structured data, data of various types and formats (explanatory texts, medical records, x-rays, etc.) and large tables (sensor generated variable records). On the other hand, NoSQL data stores are ideally suited for this kind of applications that use large amounts of disparate data. Therefore, we are convinced that a NoSQL DBMS, like MongoDB, is the most adapted system to store the medical database.

As mentioned earlier, this type of system runs on a schema-less data model, which allows users to easily add new data to their applications without first defining the schema or modifying it. However, a semantic model is still needed to write queries where table and column names are mentioned [4] by looking at how the data is structured and related to each other in the database. UML is widely regarded as a standard modeling language for the description of complex data [7].

In our view, it's important to have a precise and automatic solution that guides and facilitates the database model extraction task within NoSQL systems. For this, we propose the NoSQLToUML process presented in the section 4 that extracts the conceptual model of a database stored in MongoDB. This model is expressed using a UML class diagram.

III. RELATED WORK

The extraction of a physical model from a NoSQL database "schema less" has been the subject of several research works, this mainly for document type databases like MongoDB. Thus, a process has been proposed in [11] to extract the model from a collection of JSON documents stored on MongoDB. The model returned by this process is in JSON format; it is obtained by capturing the names of the attributes that appear in the input documents and replacing their values with their types. Attribute values can be atomic type, lists, or nested documents. In the article by [12], the authors propose another process for extracting the data model from a NoSQL database of documents type that may contain several collections. The returned result is not a unified model for the whole database, but it gives the different model versions for each collection. The extraction process is composed of two successive steps. The first one runs through the database and, for each distinct template version, generates a document in a collection called "Template". In the second step, the process provides a model of each version by instantiating the JSON meta-model. We also mention the work of [7] which proposes a process called BSP (Build Schema Profile) to classify the documents of a collection by applying rules corresponding to the requirements of the users. These rules are expressed through a decision tree whose nodes represent the attributes of the documents; the edges specify the conditions on which the classification is based. These conditions reflect either the absence or the presence of an attribute in a document or its value. As in the previous work [12], the result returned by this approach is not a unified model but a set of version models; each of them is common to a group of documents.

In [14], the authors describe the transition from a document-oriented or graph-oriented NoSQL database to a relational schema. The process groups together all documents (or objects) that have the same field names. For each class of objects thus obtained, it generates a table having as attributes the names of the fields; and as lines the values of these fields.

On the other hand, in [15], the authors propose a process of extracting the schema of a large collection of JSON documents using the MapReduce system. The Map phase consists of extracting the schema of each document from the collection by inferring pairs (field, type) from the pairs (field, value). The Reduce phase consists of unifying all the schemas produced in the Map phase in order to provide an overall schema of all the documents in the collection. In another article [16], the same authors proposed to extend this process by integrating the parameterization of the extraction at the level of the Reduce phase. Thus, the user can choose either to unify all the schemas of the documents in the collection, or to unify only the schemas having the same fields (names and types).

To our knowledge, few works have resulted in the extraction of a conceptual model for NOSQL databases. However, we can cite a proposition that deals with the graph-oriented model [13]. The authors propose a process of extraction of a conceptual model from object insertion queries and relations in a NoSQL database oriented-graphs; The proposed process is based on an MDA architecture and applies

two types of transformations. The first is to build a graph (Nodes + Edges) from Neo4j queries. The second is to extract an Entity / Association model from the graph by transforming nodes with the same label into entities and edges into associations.

Moreover, in [17], the authors proposed a process of discovery of the conceptual model (UML class diagram) from a JSON document. The process is composed of two successive steps. The first is to extract a physical model in JSON format by transforming the pairs (key, value) into pairs (key, type). The second step generates a root class CR and transforms the primitive type fields (integer, character string and boolean) into CR attributes and the fields structured into CC component classes linked to CR by composition links. Only the UML composition links are taken into account in these works.

In Table 1, we summarize the previous works by showing their main characteristics:

TABLE I COMPARATIVE TABLE OF PREVIOUS WORKS

	Database content		NoSQL system type		Links intra or inter classes		Conceptual Model Extraction	
	One class	Several classes	Document-oriented	Graph-oriented				
[11]	X		X			No		No
[12]		X	X			No		No
[7]	X		X			No		No
[14]	X		X	X		No		No
[15] , [16]	X		X			No		No
[13]		X		X	Yes		Yes	
[17]	X		X			No	Yes	

Through this state of the art, it appears that the proposals only partially answer our problematic. Indeed, to extract a physical model, the work of [11], [12], [7], [14], [15] and [16], do not consider the links between objects. Similarly, in [17], where the authors do not take into account the links between classes On the other hand, the works of [13] which focus on systems oriented -graphs, do not process structured attributes; these do not exist in this type of systems. ; this is due to the fact that the graph-oriented systems do not make it possible to declare this type of attributes.

IV. NOSQLTOUML PROCESS

The purpose of this article is to automate the process of extracting a semantic model from a NoSQL database "schema less". We generally consider four types of NoSQL databases: key / value, column, document and graph. We limit ourselves to the type document which is the most complete in terms of expression of the links (use of references and nestings). The NoSQLToUML process that we propose, automatically extracts a semantic model (UML class diagram) from a document-type NoSQL database.

As shown in Figure 1, the NoSQLToUML process consists of two modules: (1) ToPhysicalModel which corresponds to the extraction of a physical schema from a document-oriented NoSQL database and (2) ToConceptualModel which consists of to transform the physical schema returned by the first module into a conceptual schema modeled according to UML formalism.

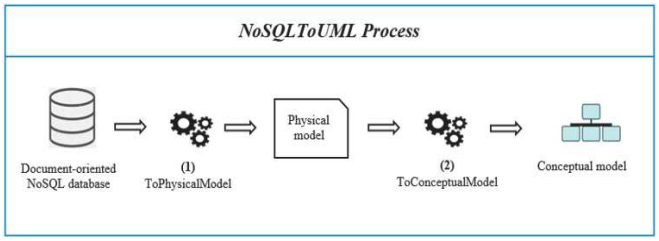

Fig. 1. Overview of *NoSQLToModel* process

To formalize and automate our process, we use OMG's Model Driven Architecture [10], which provides a formal framework for automating model transformations. The purpose of this architecture is to describe separately the functional specifications and implementation specifications of an application on a given platform [10]. For this, it uses three models representing the abstraction levels of the application. These are (1) the requirements model (CIM for Computation Independent Model) in which no IT considerations appear, (2) the independent Platform Independent Model (PIM) for the technical details. Execution platforms and (3) Platform Specific Model (PSM) specific to a particular platform. Since the input of our process is a NoSQL database and its output is a conceptual model, we retain only the PSM and PIM levels. We detail in the following the two modules composing our process.

A. ToPhysicalModel module

This article focuses on the automatic generation of NoSQL database conceptual model. To properly define this process, it is necessary to know the framework in which it will fit. It's about our approach ToPhysicalModel developed in previous work [20]. This section outlines this models transformation approach.

ToPhysicalModel is an MDA-based process that automatically generates a NoSQL physical model from a database implemented on MongoDB by executing a set of transformation rules. This approach relies on the use of the structured filed DBRef which is used when a document contains references to another documents present in a different collections [21]; it uses the value of the referenced document's _id field and the collection name (the database name is optionally). The DBRef format allowed us to provide a common semantics for representing links between documents.

ToPhysicalModel process is composed of two steps: (1) the first one browses all of the documents in a collection and extracts the schema for each one; (2) for each collection, the second step takes as input the set of schemas returned by the first step, unifies them and then provide a unique schema that describe how data is organized and related to each other in the corresponding collection .

Our process as such was limited to extracting the NoSQL database physical model in order to assist the user to express his queries. In this article, we aim to complete our approach by taking into account the generation of the database conceptual model (§ Section 4).

B. ToConceptualModel module

We present in this section the second module ToConceptualModel which represents the second step of our solution illustrated in figure 1. We begin beforehand by the formalization of the source and the target of this module. We then explain the transformations needed to move the source elements to the target elements.

1) Source: physical model:

The source of the ToConceptualModel module is the physical model obtained after running the ToPhysicalModel module. This template is stored in document collections. Each collection has a single document. This contains the model of an input collection. It is composed of a set of fields; each of them can be either primitive or complex. A primitive field is in the form of a couple (Name, Type) where Type corresponds to a data type like Integer, Boolean and String; it can be multivalued. A complex field is a structured or multivalued field of structured fields. A structured field is composed of a set of fields that, in turn, can be either primitive or complex. To express a link between collections, we used a field called DBRef, which is un standard used by the MongoDB community [21]. This one is a special case of a structured field. It is composed of two fields; one corresponds to the identifier of the referenced document and one corresponds to the name of the collection that contains the referenced document.

We present the concepts used to describe the physical model through the Ecore meta-model of Figure 2.

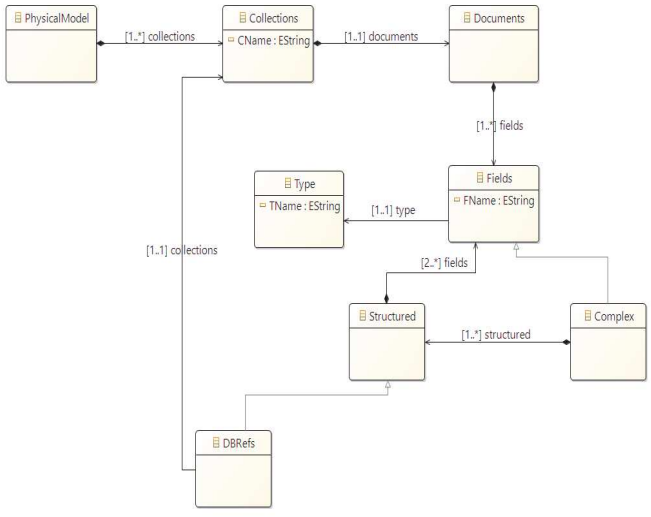

Fig. 2. Source Metamodel

2) *Target: conceptual model:*

A UML class diagram is defined by a set of object classes linked together by links. Each class has semantics and common properties; it is defined by a name, attributes, and operations. Each attribute is defined by a couple (name, type). In this article, we do not consider operations. A link is a semantic relation between two or more classes of objects. It is defined by a name, a type, and endpoints. The most common links are association, composition, aggregation and inheritance. We restrict ourselves in this article only to links of association and composition. The ends represent connections with the classes concerned by the link; they are defined by a cardinality Min and a cardinality Max. We formalize all these concepts through the Ecore meta-model of Figure 3.

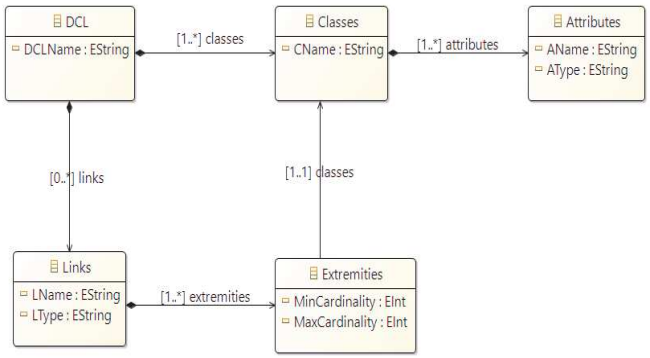

Fig. 3. Target Metamodel

3) Transformations:

After having formalized the concepts present in the source and target models of the ToConceptualModel module, we will describe the sequence of transformation rules necessary for the transition from the source elements to the target elements.

R1: Each input collection Clli is transformed into a class Ci.

R2: Each primitive field of Clli (integer, string, boolean, ... etc.) is transformed into an attribute of the same type in the corresponding class Ci; if the field is multivalued, the type of the corresponding attribute is put between two brackets [].

R3: For each collection Clli, each complex field that is not a DBRef becomes a class linked to Ci (the class corresponding to Clli) by a composition link. This one takes either the cardinality 0..1 if the input complex field is structured, or the cardinality 0..* if the input complex field is multivalued and whose values are structured.

R4: For each collection Cll1, each complex field which is a DBRef filed referencing a collection Cll2 becomes an association link connecting the two classes C1 and C2 (the classes corresponding, respectively, to Cll1 and Cll2); this association link has the same name as the input DBRef field and takes either the cardinalities 0..* next to C1 and 0..1 next to C2 if the field DBRef is monovalued, or the cardinalities 0..* next to C1 and 0..* otherwise.

The ToConceptualModel process is presented as a sequence of four steps executed one by one to generate the resulting model (UML conceptual model) from the source model (NoSQL physical model):

1: we define metamodels of the source and the target

2: we instantiate from the source metamodel, a physical model in XMI (XML Metadata Interchange) format.

3: we implement on EMF the transformation rules using the QVT language

4: To test transformations, we execute the QVT script encoded in step 3 on the source model (physical model). The returned result represents the corresponding conceptual model and is provided as an XMI file.

V. Conclusion

Our work is part of Big Data databases. They are currently dealing with the reverse engineering mechanisms of a NoSQL database "schema less" to facilitate the expression of queries.

In this article, we have proposed an automatic process NoSQLToUML to extract the conceptual model of a NoSQL database of documents type. This process, based on the Model Driven Architecture (MDA), provides a formal framework for automating model transformations. NoSQLToUML is composed of two modules; the first generates the physical model from a document-oriented database and the second transforms the physical model returned by the first module into a conceptual diagram in the form of a UML class diagram. The transition from one model to another is done by applying a sequence of transformations formalized with the QVT standard. The major contribution of our solution lies in taking into account the links of association between the classes. We have experimented our process on the case of a medical application which relates to scientific programs of follow-up of pathologies; the database is stored on the MongoDB NoSQL system.

Regarding prospects, we plan to study the update of the data model as the database is being exploited. Indeed, the data volume can reach several terabytes, the generation of the model requires the scanning of the entire database. It is therefore not possible for a user to restart the process each time he wishes to express a new request.

References

[1] 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.

[2] Abdelhedi, F., Brahim, A. A., & Zurfluh, G. (2018). Formalizing the Mapping of UML Conceptual Schemas to Column-Oriented Databases. International Journal of Data Warehousing and Mining (IJDWM), 14(3), 44-68.

[3] Bondiombouy, C. (2015). Query processing in cloud multistore systems. In BDA : Bases de Données Avancées.

[4] Budinsky, F., Steinberg, D., Ellersick, R., Grose, T. J., & Merks, E. (2004). Eclipse modeling framework: a developer's guide. Addison-Wesley Professional.

[5] Chen, CL Philip et Zhang, Chun-Yang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences, 2014, vol. 275, p. 314-347.

[6] Douglas, L., 2001. 3d data management: Controlling data volume, velocity and variety. Gartner. Retrieved, 6, 2001.

[7] Gallinucci, E., Golfarelli, M., & Rizzi, S. (2018). Schema profiling of document-oriented databases. Information Systems, 75, 13-25.

[8] Han, Jing, Haihong, E., LE, Guan, et al. Survey on NoSQL database. Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011. p. 363-366.

[9] Harrison, G. (2015). Next Generation Databases : NoSQLand Big Data. Apress.

[10] Hutchinson, J., Rouncefield, M., & Whittle, J. (2011, May). Model-driven engineering practices in industry. In Proceedings of the 33rd International Conference on Software Engineering (pp. 633-642). ACM.

[11] Klettke, M., U. Störl, et S. Scherzinger (2015). Schema extraction and structural outlier detection for json-based nosql data stores. Datenbanksysteme für Business, Technologie und Web (BTW 2015).

[12] Sevilla, Diego Ruiz, Severino Feliciano Morales, and Jesús García Molina. "Inferring versioned schemas from NoSQL databases and its applications." International Conference on Conceptual Modeling. Springer, Cham, 2015.

[13] Comyn-Wattiau, I., & Akoka, J. (2017, December). Model driven reverse engineering of NoSQL property graph databases: The case of Neo4j. In 2017 IEEE International Conference on Big Data (Big Data) (pp. 453-458). IEEE.

[14] Maity, B., Acharya, A., Goto, T., & Sen, S. (2018, June). A Framework to Convert NoSQL to Relational Model. In Proceedings of the 6th ACM/ACIS International Conference on Applied Computing and Information Technology (pp. 1-6). ACM.

[15] Baazizi, M. A., Lahmar, H. B., Colazzo, D., Ghelli, G., & Sartiani, C. (2017, March). Schema inference for massive JSON datasets. In Extending Database Technology (EDBT).

[16] Baazizi, M. A., Colazzo, D., Ghelli, G., & Sartiani, C. (2019). Parametric schema inference for massive JSON datasets. The VLDB Journal, 1-25.

[17] Generate Test Data (2018) http://www.convertcsv.com/generate-test-data.htm Online; 5 July 2018.

[18] JSON Generator (2018) http://www.json-generator.com/. Online; 5 July 2018.

[19] Ait Brahim, A., Tighilt Ferhat, R., Zurfluh, Z, MDA Process to Extract the Data Model from Document-oriented NoSQL Database, ICEIS 2019.

[20] MongoDB, https://www.mongodb.com/fr Online; 5 July 2018.