International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I5P215

Volume 71 Issue 5, 146-155, May 2023

Original Article

Towards the Generation of a PSM Model from a PIM Model, Integration of the MDA Approach in NoSQL Databases, the Case of Document-Oriented NoSQL Platforms

Aziz Srai[1], Fatima Guerouate[2]

[1]*ERCI2A, FSTH, Abdelmalek Essaadi University, Tetouan, Morocco*
[2]*LASTIMI Laboratory, Superior School of Technologies of Sale, Mohammadia School of Engineering, Mohamed V University City of Rabat, Morocco*

[1]*Corresponding Author : a.srai@uae.ac.ma*

Received: 22 February 2023 Revised: 02 April 2023 Accepted: 28 April 2023 Published: 25 May 2023

Abstract *- Model-driven engineering has allowed several significant improvements in the development of complex systems by allowing one to focus on a more abstract concern than classical programming. It is a form of generative engineering in which all or part of an application is generated from models. A model is an abstraction, a simplification of a system that is sufficient to understand the modeled system and answer the questions that arise about it. A system can be described by different related models each other. The key idea is to use as many different modeling languages as system development's chronological or technological aspects require—two major concepts of model-driven engineering, metamodeling and model transformation. Model transformation makes models operational for code generation, documentation and testing, validation, verification, execution, and so on. In the same context, the massive evolution of data has generated a new notion for the processing of these data; it is the notion of NoSQL. NoSQL databases are a new generation of databases that allow the processing and exploitation of massive data. In the work presented in this article, we will try to combine between the two, the MDA approach and NoSQL databases. We will generate a PSM model for NoSQL databases based on the document to validate the validity and applicability of the MDA approach with NoSQL models. We consider a case study of a simple class diagram sufficient to demonstrate the approach.*

Keywords *- Big data, MDA approach, Model programming, NoSQL, Qvt.*

1. Introduction

In order to achieve platform-independent reuse, the OMG subsequently proposed the MDA architecture (Model Driven Architecture). With the MDA approach, the development of software platforms is centered on technology-independent models (the PIM models - platform Independent Model), which are then refined into models containing technical details (the PSM models - Platform Specific Model). By expressing with models the different levels of abstraction of a system, the MDA architecture has enabled reuse in the most upstream phases of development, such as analysis and design. MDE (Model Driven Engineering) then generalized the approach beyond architectural issues. This new approach represents all dimensions of engineering, its products and processes through models. In addition to models, this engineering uses a second form of artifacts. These are model transformations. These are used to manipulate the models, for example, modifying them, representing them in another formalism, generating code, etc. the reuse of transformations makes it possible to respond to complex tasks. It consists of creating transformations from existing ones, adding or modifying transformation rules and combining transformations. Typically, combining transformations is done by chaining them.

In another context, we have noticed the increase in the volume of data reaching critical proportions; this massive data is mainly processed through Relational Database Management Systems (RDBMS). These approaches prove to be difficult to extend to big data, and this is the reason why a new notion has emerged, it is the NoSQL model or NoSQL databases. In the work presented in this article, we will try to combine between the two, the MDA approach and NoSQL databases. We will generate a PSM model for Document-based NoSQL databases to validate the validity and applicability of the MDA approach with NoSQL models. We consider a case study of a simple class diagram sufficient to demonstrate the approach.

2. Literature Review

In the literature, several research projects have been proposed as part of the integration of the MDA approach in NoSQL databases. In [Chevalier, 2015], the authors defined a set of rules to map a star schema into two NoSQL models: column-oriented and document-oriented. Other studies [Li et al., 2010] and [Vajk et al., 2013] investigated 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 correspondence between a relational model and a document-oriented model using MongoDB. [Li et al., 2014] propose a method based on the MDA process to transform the UML class diagram into a column-oriented model specific to HBase. [Gwendal et al., 2016] describe the correspondence between a UML conceptual model and graph databases via an intermediate graph metamodel. In this work, the transformation rules are specific to graph 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 strongly connected. To our knowledge, no work has presented a global study to transform a source model (uml diagram) into a target model (graph-oriented NoSQL databases), i.e. the key-value generation of NoSQL database via an MDA approach. According to the analysis of the works cited, we have found that a majority of authors do not invoke an interest in applying the MDA approach on NoSQL platforms via a transformation from a PIM model to a model PSM or through code generation through a PSM-to-code transformation. To our knowledge, we are the first authors to have proposed a total generation of code for NoSQL platforms to use models independent of all implementation platforms. No work has presented a global study to transform a source model (uml diagram) into a target model (Document-oriented NoSQL databases), i.e. a generation of a NoSQL Document database via an MDA approach.

3. Research Methodology

3.1. Model-Driven Engineering

Following the object approach of the 80s and its "everything is object" principle, software engineering is now moving towards model-driven engineering and the "everything is model" principle. This new approach can be considered both in continuity and rupture with previous work. First of all, continuity because it is the object technology that triggered the evolution towards models. Indeed, once acquired the design of computer systems in the form of objects communicating with each other, the question arose of classifying them according to their different origins (business objects, techniques, etc.). Model-driven engineering, therefore, aims, in a more radical way than could be the approaches of patterns and aspects, to provide a large number of models to separately express each of the concerns of users, designers, architects, etc. It is through this fundamentally different basic principle that model-driven engineering can be considered to break with the work of the object approach. While the object approach is based on two essential relationships, "InstanceFrom" and "InheritFrom", model-driven engineering is based on another set of concepts and relationships. The central concept of MDI is the notion of a model for which there is currently no universal definition.

3.2. The MDA Approach

The consensus on UML was decisive in this transition to model-based production techniques. After the acceptance of the key concept of metamodel as a model description language, many metamodels have emerged in order to each bring their specificities in a particular domain (software development, data warehouse, development process, etc.). Faced with the danger of seeing this great variety of metamodels emerge independently and in an incompatible manner, there was an urgent need to provide a general framework for their description. Therefore, the logical answer was to offer a language for defining metamodels, which itself took the form of a model: the metametamodel MOF (Meta-Object Facility). As a model, it must also be defined from a modeling language. To limit the number of levels of abstraction, it must then have the property of metacircularity, i.e. the ability to describe itself.

3.3. Metametamodel

A metametamodel is a model that describes a metamodeling language, i.e. the modeling elements necessary for the definition of modeling languages. He also has the ability to describe himself. It is on these principles that the organization of the OMG modeling is based, generally described in a pyramidal form. The real world is represented at the pyramid's base (level M0). The models representing this reality constitute level M1. The metamodels allowing the definition of these models (e.g. UML) constitute level M2. Finally, the metametamodel, unique and metacircular, is represented at the top of the pyramid (level M3).

3.4. Technical Area

A technical space is the set of tools and techniques resulting from a pyramid of metamodels whose top is occupied by a family of similar metametamodels. The OMG defined MDA (Model Driven Architecture) in 2000 to promulgate good modeling practices and fully exploit the benefits of models. In 2003, members adopted the latest specification version giving a detailed architecture definition. This approach aims to highlight the intrinsic qualities of the models, such as durability, productivity and consideration of execution platforms. The MDA includes for this the definition of several standards, in particular UML, MOF and XMI. The key and initial principle of MDA consists in relying on the UML standard to describe models separately for the different phases of the development cycle of an application. More specifically, the MDA advocates the development of models.

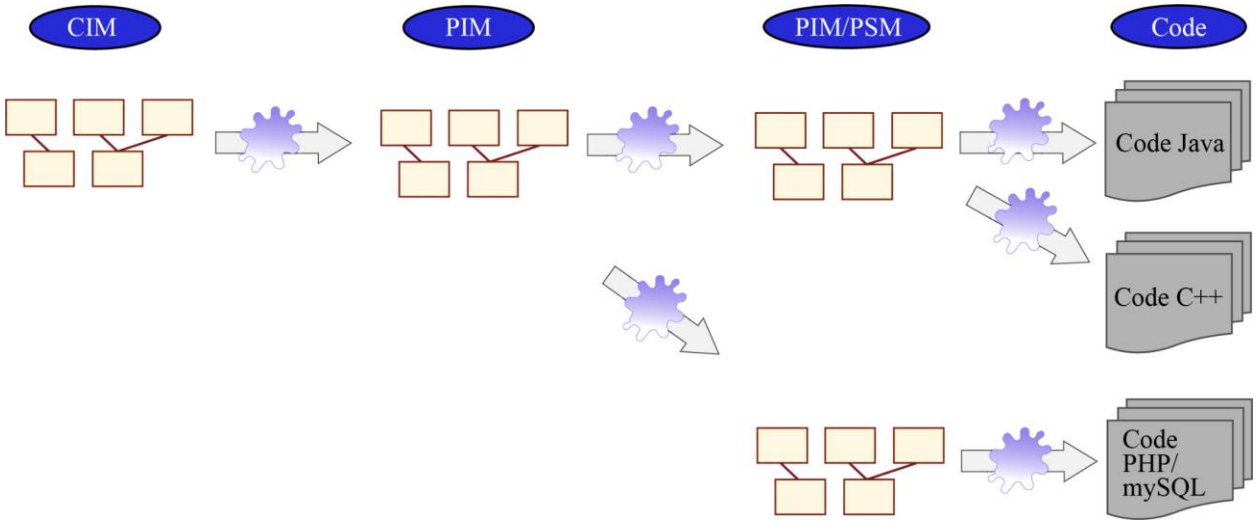


Fig. 1 The MDA lifecycle

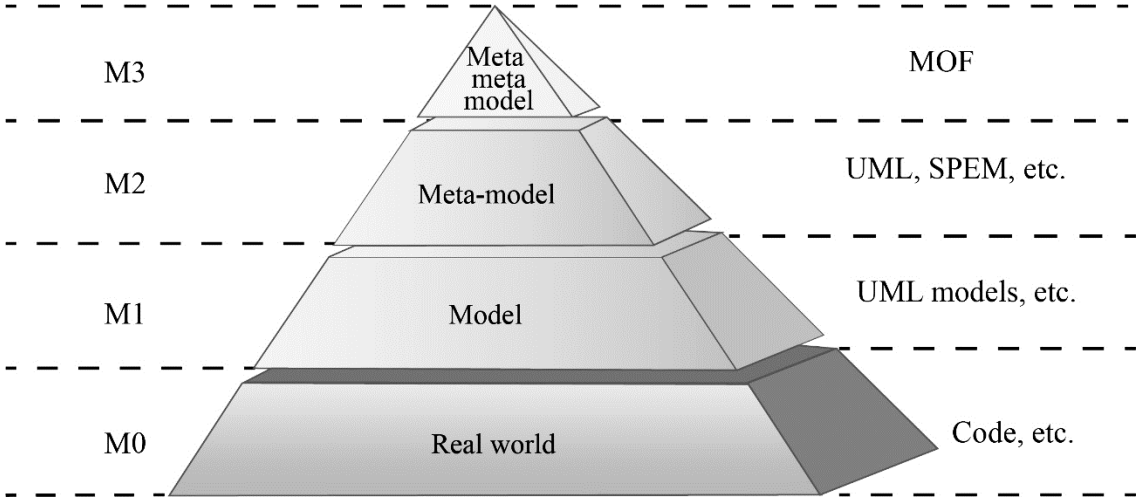


Fig. 2 OMG modeling pyramid

Requirements (Computation Independent Model – CIM) in which no IT consideration appears, analysis and design (Platform Independent Model – PIM), code (Platform Specific Model – PSM). The major objective of the MDA is the development of durable models (PIM), independent of the technical details of the execution platforms (J2EE, .Net, PHP, etc.), to allow the automatic generation of all of the code models (PSM) and to obtain a significant gain in productivity.

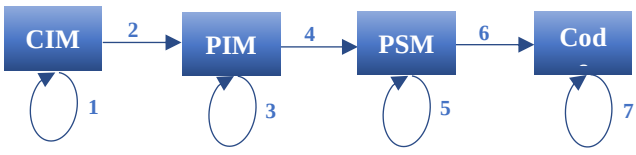


Fig. 3 Transformation of models in the MDA approach

3.5. Dedicated Modeling Languages

In the same way that the arrival of object-oriented programming did not invalidate the contributions of structured programming, model-driven development does not contradict the contributions of object technology. Therefore, it is important not to consider these solutions as antagonistic but complementary. However, a point of divergence between these two approaches concerns the integration of paradigms. Initially, object technology was also intended as an integration technology because it was theoretically possible to uniformly represent processes, rules, functions, etc., by objects. Today, we return to a less hegemonic vision where the different programming paradigms coexist without giving more importance to one or the other. An important point is then to separate the model-driven engineering approaches from the UML formalism clearly and from the use made of them in the MDA. Indeed, not only is the scope of model-driven engineering broader than that of UML, but the vision of model-driven engineering is also very different from that of UML, sometimes even in contradiction. UML is a fairly monolithic standard obtained by maximum consensus, the scope of which must be reduced or extended using mechanisms such as profiles. These mechanisms do not all have the desired precision and sometimes lead to dangerous

contortions to "stay" in the UML world. On the contrary, model-driven engineering favors the definition of modeling languages dedicated to a particular domain (Domain Specific Modeling Languages – DSML), thus offering users concepts specific to their profession and which they have mastered. These languages are generally small in size and must be easily manipulated, transformed, combined, etc.

3.6. Standards and Languages for Model Transformation

Many languages are currently available for writing generation model transformations. We first find generalist languages that rely directly on the abstract representation of the model. We will quote, by example, the EMF API, which, coupled with the Java language, makes it possible to manipulate a model in the form of a graph. In this case, it is up to the programmer to search for information in the model, to explain the order of application of the rules, to manage the target elements built, etc.. to abstract the definition of model transformations and make the implementation details transparent; the idea was to define DSMLs dedicated to model transformation. This approach is then based on the definition of a metamodel dedicated to model transformation and tools allowing to execute transformation models. We will cite, for example, ATL (ATLAS Transformation Language) that we use throughout this thesis. It is a hybrid language (declarative and imperative) which allows the definition of a transformation from model to model (called Module) in the form of a set of rules. It also allows you to define model-to-text type transformations (called Query). A transformation takes as input a set of models (described from metamodels in Ecore or KM3). In order to provide a normative framework for the implementation of the different languages dedicated to model transformation, the OMG has defined the QVT (Query/View/Transformation) standard. QVT's metamodel is MOF compliant, and OCL is used for model navigation. The metamodel shows three sub-languages for the transformation of models, characterized by the paradigm implemented to define the transformations (declarative, imperative and hybrid). The Relations and Core languages are both declarative but placed at different levels of abstraction. One of the goals of Core is to provide a basis for specifying the semantics of Relations. The semantics of Relations is given as a transformation from Relations to Core. Defining a declarative solution to a given transformation problem is sometimes difficult. To address this issue, QVT offers two mechanisms to extend Relations and Core: a third language called Operational Mappings and a mechanism for invoking transformation features implemented in an arbitrary language (black box or black box). Operational Mappings extend Relations with imperative constructs and OCL constructs with side effects.

3.7. NoSQL Models

The term NoSQL was first used in 2009. This term refers to a new performance-based approach to databases capable of handling a large amount of information, ensuring high service availability and having good scalability. To manage scalability. NoSQL adopts a non-relational approach; its development has been marked by the abandonment or compromise made on the ACID properties of relational databases, qualified so far as an obstacle to horizontal scalability. A significant number of NoSQL solutions have developed. There are different ways to structure data, but all the solutions developed can be divided into four models, the key-value-oriented model, column-oriented model, document-oriented model and graph-oriented model. In the beginning, these approaches were developed by relaxing the relational principles. NoSQL is currently considered a complementary solution; an important difference concerns the data model adopted by each of these solutions.

3.8. Key-Value Oriented Model

The key-value model considers each record as a key associated with a value. This approach is like a two-column associative array where the key uniquely identifies the associated value. Unlike the relational model, the size and type of the value are not determined beforehand. The case of the key-value model is more flexible; the value of the key can change from one record to another, can each line has a different type and size. A second characteristic of the key-value model lies in the modularity of the keys, which can be generated automatically or constructed according to a certain logic. How the key can be constructed is left up to you; the developer can imagine various key generation methods to facilitate his search.

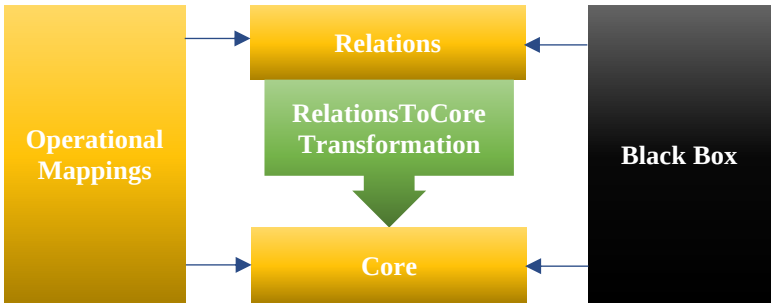


Fig. 4 Relationships between QVT metamodels

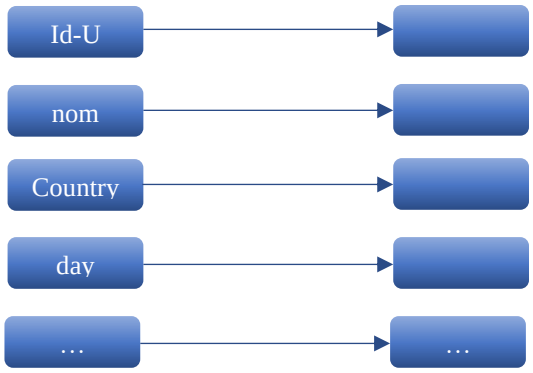


Fig. 5 Principle of the key-value model

Organizing data into key-value pairs allows for good performance. The keys are natively indexed. Communication with the database is limited to basic operations designated by the acronym CRUD (create, read, update, delete). On the other hand, search performance is impacted based on other criteria, particularly parts of the value. The key-value model is the fundamental NoSQL model. It has a simple structure that allows it to gain significant performance. However, it does not allow fine manipulation of the value. This limitation has motivated the development of column- and document-oriented models, which can be considered an evolved form of the key-value model. These two models introduce a structuring of value according to orthogonal principles. The value can thus be either atomic or compound. These models are distinguished by the storage of data which is carried out either by document (horizontal) or by line broken down into families of columns (vertical). A fourth NoSQL model is based on the graph-oriented model. It is characterized by a structure based on the theory of labeled graphs.

3.9. Document-Oriented Model

Following the same principle as the key-value-oriented model, the document-oriented model consists of key-value pairs. The key identifies a structured value called a document. A document is considered a hierarchy of elements that can be either atomic or compound values (multiple atomic values or nested documents). The nesting level is not limited. A value can also be a reference to another document. All the documents are contained in a collection corresponding to the notion of a relational table and follow horizontal physical storage. In the document-oriented model, a document is a structure readable by the NoSQL engine. It is defined in a tagged textual format, usually JSON (JavaScript Object Notation) format. It facilitates the representation of structured data, especially in a hierarchical way. Other representation formats are possible, such as XML.

In the document-oriented NoSQL approach, the schema is not established in advance, each document can have its own structure, and we speak of dynamic schema. Natively, the recording of values in a document is not subject to any system control.

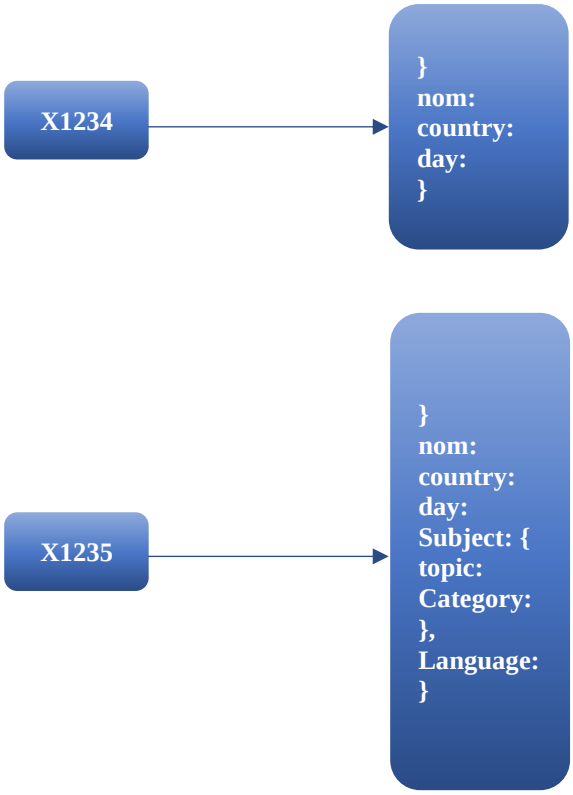


Fig. 6 Principle of the document-oriented model

However, keeping a common minimal structure to facilitate data manipulation is preferable. Unlike the key-value model, it is possible to manipulate the elements that make up a given value directly. For example, it becomes possible to manipulate a subpart of a document representing a user, such as his name or his address, without having to extract the whole value. In addition, the document-oriented model allows extended indexing to other attributes (other than the key), which can improve query performance.

3.10. Column-Oriented Model

The column-oriented model is the second evolved form of the key-value model. In relational databases, the relationship schema determines the data structure in advance, with a limited number of columns, each similar for all records (“tuples”). The column-oriented model provides a flexible schema (untyped columns) that can vary between each record (each row). The flexibility of a column-oriented NoSQL database makes it possible to manage the absence of certain columns between the different rows of the table. A column-oriented database is a set of tables defined row by row but whose physical storage is organized vertically by a group of columns called “column families”. A column family can contain a very large number of columns. The number of columns can vary from row to row; each column family is like a key-value set where the key is seen as a column associated with a value.

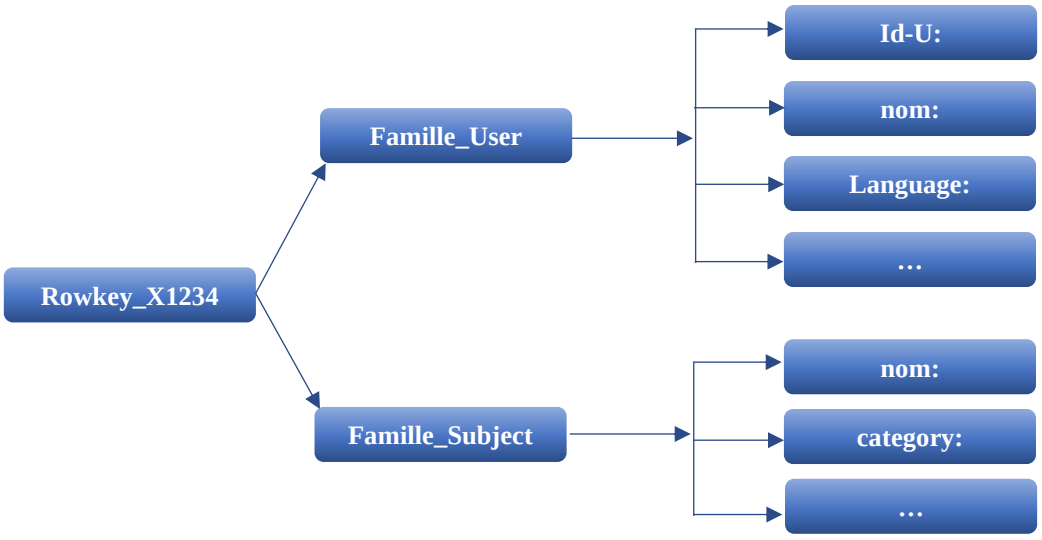


Fig. 7 Principle of the column-oriented model

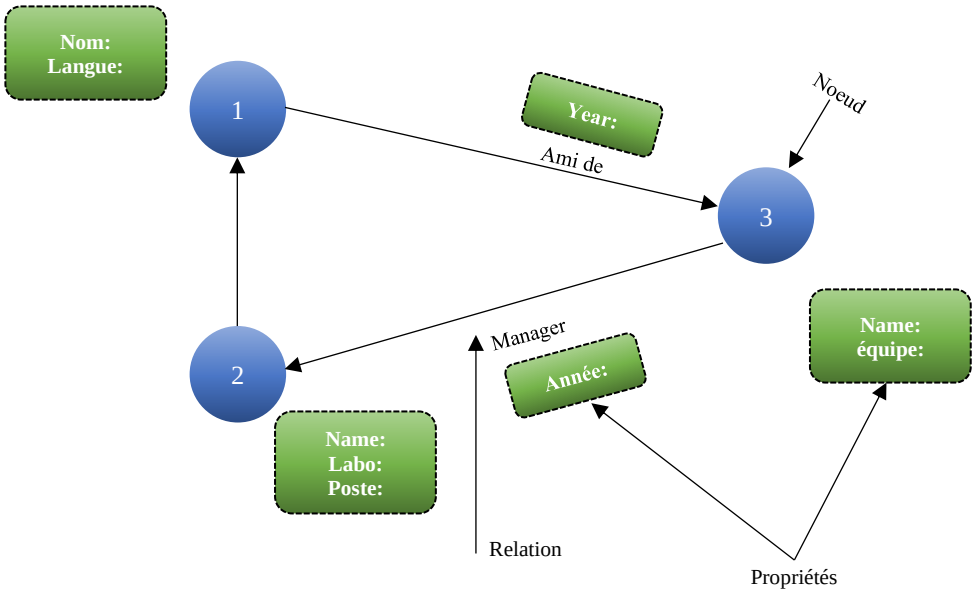


Fig. 8 Principle of the graph-oriented model

3.11. Graph-Oriented Models

The graph-oriented model is based on graph theory. The graph-oriented model is based on three notions; node, relation and property. Each node has properties. Relations connect nodes and optionally have properties. This type of approach facilitates navigational queries between nodes by following the relationships that connect them: each node has a physical pointer to neighboring nodes allowing the fast local search.

The structure of the graph-oriented model is very suitable for responding to issues such as the management of a company's social network or any other storage requiring the traversal of graphs. The flexibility of schema also characterizes the graph-oriented model; it is unnecessary to create a schema beforehand for the nodes and the relations.

4. Results and Discussions

In our MDA approach, we opted for modeling approaches to generate the document-oriented NoSQL database. These approaches require a source metamodel and a target metamodel. In this section, we present the different meta-classes that make up the UML class diagram source metamodel and the document-oriented NoSQL target metamodel—the process of transforming the UML source model into a document-oriented target model.

Fig. 9. Illustrates the simplified UML source metamodel based on packages, including operations, associations and classes. Those classes are composed of properties with parameters.

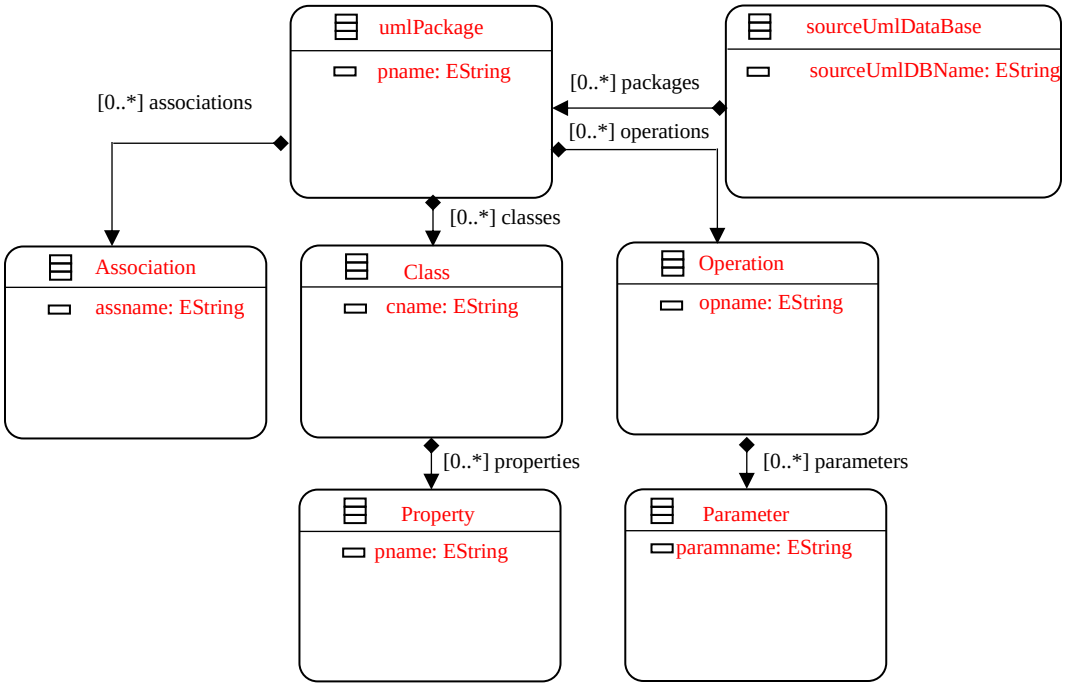


Fig. 9 Simplified UML source metamodel

Fig. 10. Illustrates the simplified Document target metamodel:

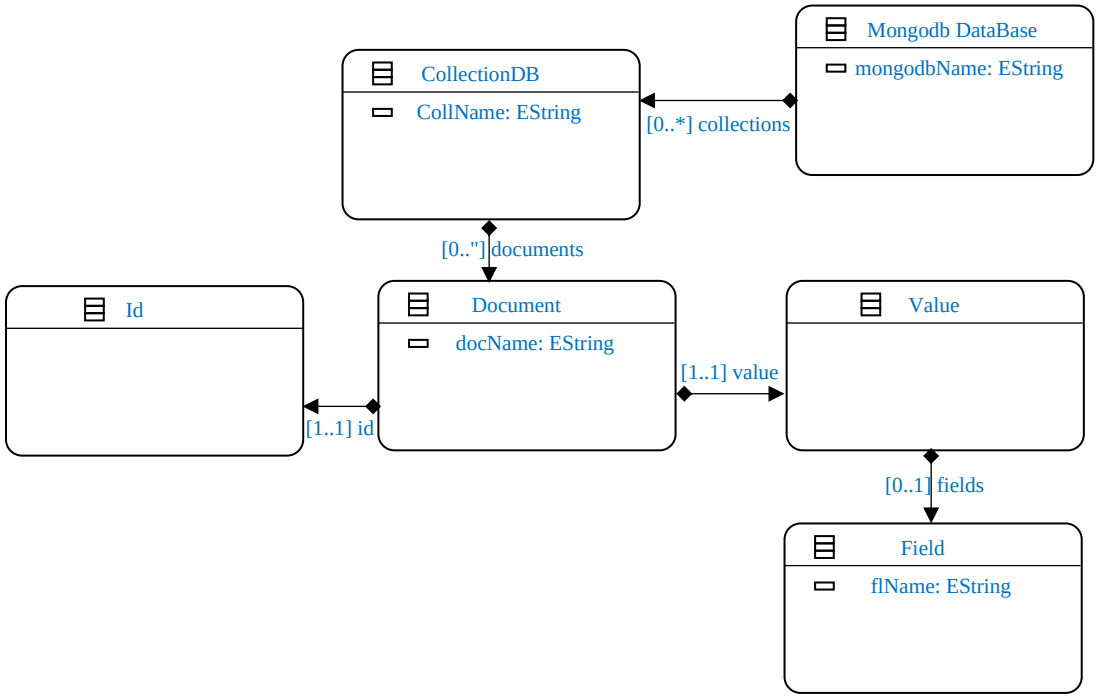
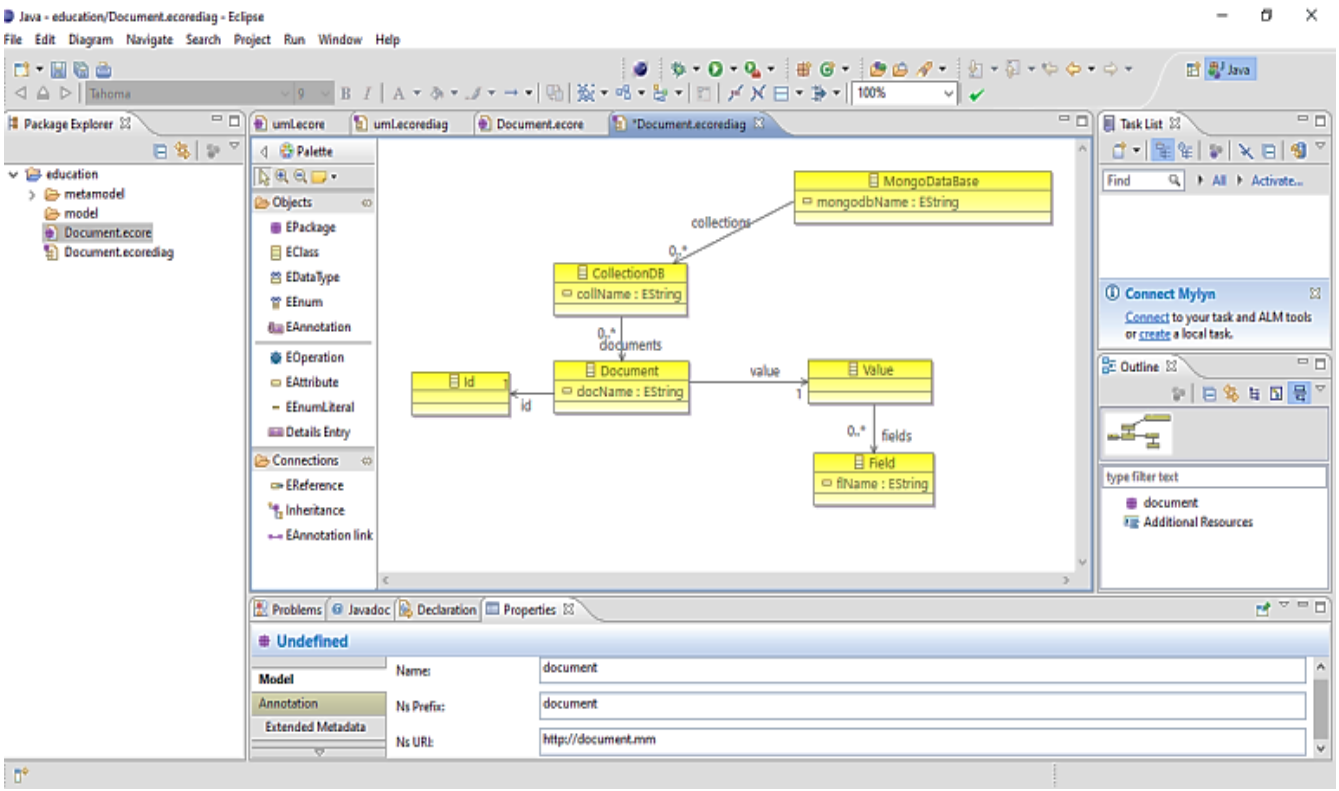


Fig. 10 Simplified document target metamodel

Fig. 11. Illustrates the transformation rules M2M and M2T:

```
modeltype umlDocument uses umlDocument("http://umlDocument.mm");
modeltype MongoDB uses MongoDB("http://MongoDB.mm");

transformation umlToDocumentDB(in source:umlDocument, out target:MongoDB);

main() {

      source.rootObjects()[sourceUmlDataBase]->map UmlPackageToDocumentDB();
}
mapping sourceUmlDataBase :: UmlPackageToDocumentDB() : MongodbDataBase
      {
      result.mongodbName := self.sourceUmlDBName;
     result.collections+=self.packages-> map PackageToCollection();
}

mapping umlPackage :: PackageToCollection() :   CollectionDB
      {
     result.collName := self.pname;
     result.documents+=self.classes-> map ClassToDocument();
}

mapping Class :: ClassToDocument() :    Document
      {
     result.docName := self.cname;
     result.value.fields+=source.rootObjects()[Property]->map PropertyToField();
}

 mapping Property :: PropertyToField() :    Field
      {
     result.flName := self.propname;

}
```



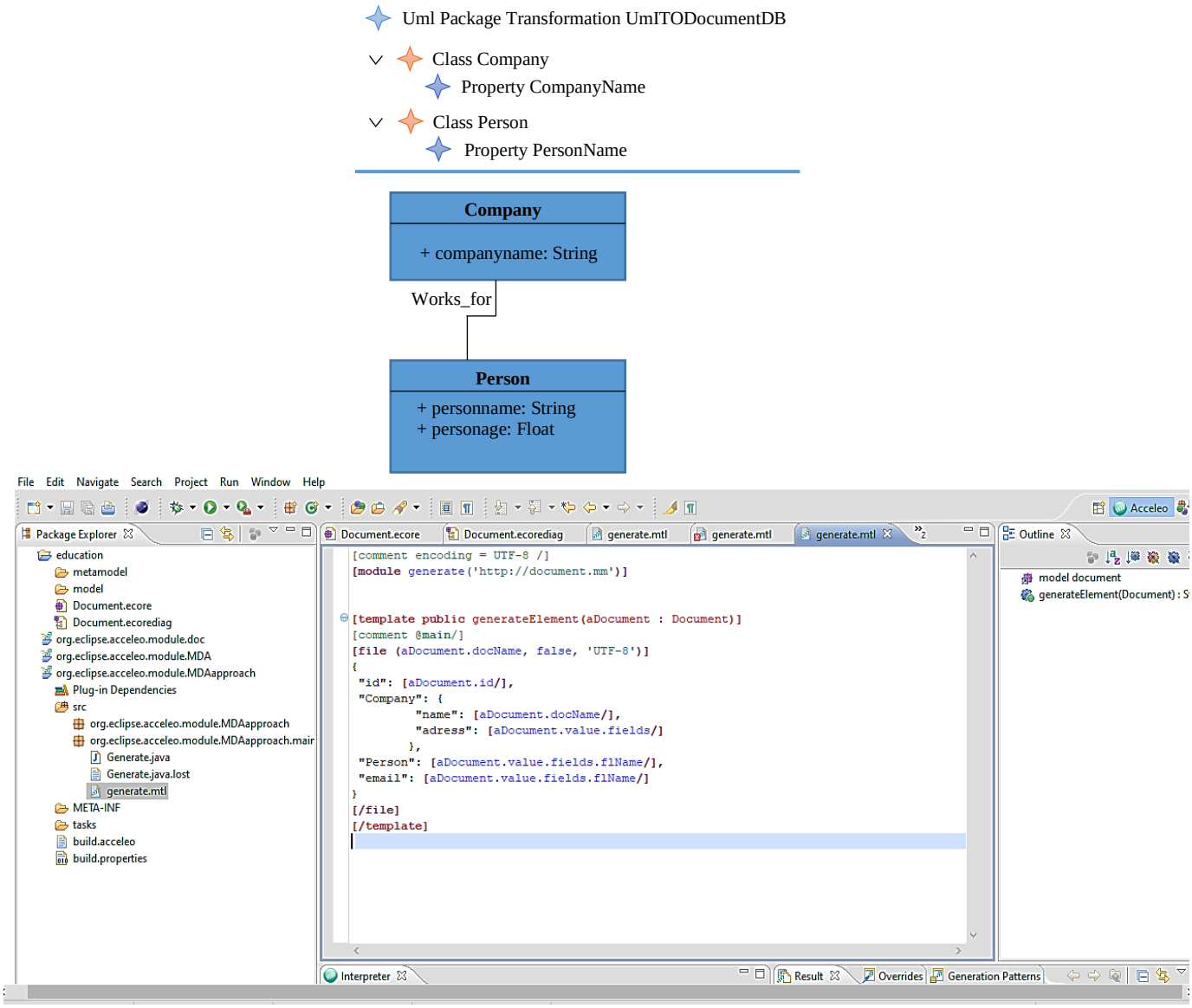


Fig. 11 Simplified document target metamodel

Fig. 12 Simplified document-Oriented MongoDB PSM

To validate our transformation rules, we conducted several tests. After applying the transformation to the UML source model, we generated the document-oriented PSM target model (see Fig. 12).

5. Conclusion

In this work, we have proposed an MDA approach to transform a UML class diagram into a document-oriented database. The transformation rules were developed using the QVT model transformation language. This work should be extended to allow the generation of other NoSQL solutions, such as key-value-oriented and column-oriented solutions. Through this work, we have validated the validity and performance of the MDA approach. Today, we can say that the application of such an approach in the context of Big Data is very important and also a current research axis.

References

[1] Aziz Srai et al., "Generated PSM Web Model for E-learning Platform Respecting n-tiers Architecture," *International Journal of Emerging Technologies in Learning*, vol. 12, no. 10, pp. 212-220, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[2] Aziz Srai et al., "MDA Approach for EJB Model," *6th IEEE International Conference on Multimedia Computing and Systems*, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[3] Nandula Rohan Kausik, Nandula Nikhil Kartik, and G. Soma Sekhar, "Personal Expense Tracker utilizing Amazon Web Services," *International Journal of Computer Trends and Technology*, vol. 70, no. 11, pp. 8-14, 2022. [CrossRef] [Publisher Link]

[4] Vahid Gharavi, Ali Mesbah, and Arie van Deursen, "Modelling and Generating AJAX Applications: A Model-Driven Approach," *Proceeding of the 7th International Workshop on Web-Oriented Software Technologies*, New York, USA. [Google Scholar] [Publisher Link]

[5] Dr.D.Shravani, "Research Methodology on Security Engineering for Web Services Security Architectures Extended for Integration of Cloud, Big Data and IoT," *SSRG International Journal of Computer Science and Engineering*, vol. 3, no. 6, pp. 18-24, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[6] J. Bezivin et al., "Applying MDA Approach for Web Service platform," *Proceedings Eighth IEEE International Enterprise Distributed Object Computing Conference*, pp. 58-70, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[7] Zuzana Bizonova, Daniel Ranc, and Matilda Drozdova, "Model Driven e-Learning Platform Integration," *Proceedings of the EC-TEL 2007 PROLEARN Doctoral Consortium*, Crete, Greece, 2007. [Google Scholar] [Publisher Link]

[8] G. Muneeswari et al., "Urban Computing: Recent Developments and Analytics Techniques in Big Data," *International Journal of Engineering Trends and Technology*, vol. 70, no. 7, pp. 158-168, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Xiao Cong et al., "A Model Driven Architecture Approach for Developing E-Learning Platform," *International Conference on Technologies for E-Learning and Digital Entertainment*, vol. 6249, pp. 111-122, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[10] Aziz Srai, Fatima Guerouate, and Hilal Drissi Lahsini, "Generated Psm Multi-Layered Model Using Mda Approach," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 4, 2019. [Google Scholar] [Publisher Link]

[11] Imane Essebaa, and Salima Chantit, "QVT Transformation Rules to Get PIM Model from CIM Model," *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pp. 195-207, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[12] Julia N. Korongo, Samuel T. Mbugua, and Samuel M. Mbuguah, "A Review Paper on Application of Model-Driven Architecture in Use-Case Driven Pervasive Software Development," *International Journal of Computer Trends and Technology*, vol. 70, no. 3, pp. 19-26, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Yassine Rhazali, Youssef Hadi, and Abdelaziz Mouloudi, "Model Transformation with ATL into MDA from CIM to PIM Structured through MVC," *Procedia Computer Science*, vol.83, pp. 1096-1101, 2016. [CrossRef] [Google Scholar] [Publisher Link].

[14] Sarra Roubi, Mohammed Erramdani, and Samir Mbarki, "Model Driven Architecture as an Approach for Modeling and Generating Graphical User Interface," *Proceedings of the Mediterranean Conference on Information & Communication Technologies*, pp. 651-656, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[15] M.Upendra Kumar, "Theoretical Analysis on Agile Security Architecture Model," *International Journal of Computer & Organization Trends*, vol. 4, no. 5, pp. 39-42, 2014. [CrossRef] [Publisher Link]

[16] Mbarki, S. and Rahmouni, M. "Combining UML Class and Activity Diagrams for MDA Generation of MVC 2 Web Applications," *International Review on Computers and Software*, vol. 8, no. 4, pp. 949-957, 2013. [Google Scholar] [Publisher Link]

[17] Frédéric Jouault, and Ivan Kurtev, "Transforming Models with ATL," *Proceedings of MoDELS 2005 Workshops*, Springer-Verlag Berlin Heidelberg, vol. 3844, pp. 128 –138, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[18] Krzysztof Czarnecki, and Simon Helsen, "Classification of Model Transformation Approaches," *Proceedings of the 2nd OOPSLA'03 Workshop on Generative Techniques in the Context of MDA*, Anaheim, 2003. [Google Scholar] [Publisher Link]