# Toward Automatic Generation of Column-Oriented NoSQL Databases in Big Data Context

https://doi.org/10.3991/ijoe.v15i09.10433

Redouane Esbai (✉), Fouad Elotmani, Fatima Zahra Belkadi
Mohammed First University, Oujda, Morocco
es.redouane@gmail.com

**Abstract**—The growth of application architectures in all areas (e.g. Astrology, Meteorology, E-commerce, social network, etc.) has resulted in an exponential increase in data volumes, now measured in Petabytes. Managing these volumes of data has become a problem that relational databases are no longer able to handle because of the acidity properties. In response to this scaling up, new concepts have emerged such as NoSQL. In this paper, we show how to design and apply transformation rules to migrate from an SQL relational database to a Big Data solution within NoSQL. For this, we use the Model Driven Architecture (MDA) and the transformation languages like as MOF 2.0 QVT (Meta-Object Facility 2.0 Query-View-Transformation) and Acceleo which define the meta-models for the development of transformation model. The transformation rules defined in this work can generate, from the class diagram, a CQL code for creation column-oriented NoSQL database.

**Keywords**—SQL relational database, NoSQL, MDA, The transformation rules, Big Data, CQL

## 1 Introduction

In recent years, the world of data storage is changing rapidly. New technologies and new actors are settling when the old ones make the move. This scientific revolution that has invaded the world of information and the Internet has imposed new challenges on researchers in recent years and has led them to design new tools for specific storage and manipulation. The development of these tools is generating a growing interest among scientific and economic actors to offer them the possibility of managing all these masses of data with reasonable response times. Big Data is correlated between four notions generally grouped under the acronym "4V", namely: Volume, Variety, Velocity and Variability [1].

The Big Data issues are part of a complex context, faced two major concerns:

- Implementation of new mass storage solutions
- Capture information at high speed and if possible in real time

Our focus in this paper is only on Big Data storage. Using relational databases proves to be inadequate for all applications, particularly ones involving large volumes of data. In this context, NoSQL databases offer new storage solutions in large-scale environments, replacing many traditional database management systems [2]. The key feature of NoSQL databases is that they are schema-less, meaning that data can be inserted in the database without upfront schema definition. Nevertheless, there is still a need for a semantic data model to define how data will be structured and related in the database [3]; it is generally accepted that UML meets this requirement [4].

Nowadays, many organizations have begun to consider MDA as an approach to design and implement enterprise applications. In this context the Model Driven Engineering provides abstraction through high level models and allows the use of modeling languages to automate the generations of applications from the model. The interest for the Model Driven Engineering (MDE) was increased towards the end of the last century, when the Object Management Group had made public its initiative Model Driven Architecture (MDA) like as restriction of the MDE [5].

Therefore, Abdelhedi et al. [6] explain how to store Big Data in NoSQL databases and they propose a MDA-based approach that transforms an UML conceptual model describing Big Data into a column-oriented NoSQL model. The result of this transformation is PSM model.

This paper aims to rethink the work presented in [6]. However, we develop the transformation rules using the MOF 2.0 QVT standard to generate a file which contains a code for creation a column-oriented NoSQL model. Our approach includes UML modeling and automatic code generation using Acceleo with the aim to facilitate and accelerate the creation of column-oriented NoSQL database.

This paper is organized as follows: related works are presented in the second section, the third section defines the MDA approach, and the fourth section presents the NoSQL and its implementation as a database, column-oriented in this case. In the fifth section, we present the source and target meta-models. In the sixth section, we present the transformation process M2M and M2T from UML class diagram model to the column-oriented NoSQL database. The last section concludes this paper and presents some perspectives.

## 2 Related Works

Many researches on MDA and the process of transforming relational databases into a NoSQL model have been conducted in recent years. The most relevant are [3, 6-10]:

Chevalier et al. [7] defined rules to transform a multidimensional model into NoSQL column-oriented and document-oriented models. The links between facts and dimensions have been converted using imbrications. Although the transformation process proposed by authors start from a multidimensional model, it contains facts, dimensions and one type of links only.

Gwendal et al. [3] describe the transformation from an UML conceptual model into a graph databases via an intermediate graph meta-model. These transformation rules

are specific to graph databases used as a framework for storing, managing and querying complex data with many connections.

Li et al. [8] propose a MDA approach to transform UML class diagram into HBase. After building the meta-models of UML class diagram and HBase, the authors have proposed mapping rules to realize the transformation from the conceptual level to the physical level. These rules are applicable to HBase only. Another works followed the same logic and have been the subject of a work Vajk et al. [9]. The authors propose a mapping from a relational model to document-oriented model using MongoDB.

The purpose of the work [10] presented by Abdelhedi et al. is to implement a conceptual model describing Big Data into NoSQL database and they choose to focus on column-oriented NoSQL model.

This paper aims to rethink and to complete the work presented by Abdelhedi et al. [6, 10], by applying the standard MOF 2.0 QVT and Acceleo to develop the transformation rules aiming at automatically generating the creation code of column-oriented NoSQL database. It is actually the only work for reaching this goal.

## 3 Model Driven Architecture (MDA) Approach

In November 2000, OMG, a consortium of over 1 000 companies, initiated the MDA approach. The major objective of MDA [5] is to develop sustainable models; those models are independent from the technical details of platforms implementation (JavaEE, .Net, PHP or other), in order to enable the automatic generation of all codes and applications leading to a significant gain in productivity. MDA includes the definition of several standards, including UML [12], MOF [13] and XMI [14].

The key principle of MDA is the use of models at different phases of application development. Specifically, MDA advocates the development of requirements models (CIM), analysis and design (PIM) and code (PSM).

### 3.1 The Transformations of MDA Model

The MDA identifies several transformations during the development cycle [15]. It is possible to make three different types of transformations: CIM to PIM, PIM to PSM and PSM to Code.

Currently, the models' transformations can be written according to three approaches: The approach by Programming, the approach by Template and the approach by Modeling.

- **Approach by programming:** using the object oriented programming languages such as Java, to write computer programs that are unique to manipulate models.
- **Approach by Modeling:** It consists of applying concepts from model engineering to models' transformations themselves. The objective is modeling a transformation, to reach perennial and productive transformation models, and to express their independence towards the platforms of execution.

- **Approach by template:** Consists of taking a "template model", canvas of configured target models, these settings will be replaced by the information contained in the source model. This approach requires a special language for defining model template.

In this paper we chose two types of transformation, we start with the transformation PIM to PSM using the approach by modelling. This type of transformation will allow us to automatically generate a column-oriented NoSQL model from an UML model. The second transformation is of type PSM to Code using the approach by template with Acceleo to develop the transformation rules aiming at automatically generating the creation code of column-oriented NoSQL database.

### 3.2 The Elaborationist Approach

After analyzing the current state of the MDA implementation, it is reasonable to say that there are two main responses to the OMG's definition: the elaborationist and translationist approach [16].

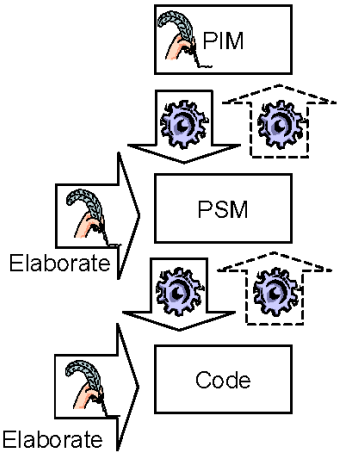


**Fig. 1.** Elaborationist approach [16]

The elaborationist approach is the one used in the present paper. The main advantage of MDA in the development of column-oriented NoSQL databases is the automation. This way, to demonstrate the automation support provides by our MDA approach, we are using the "Elaborationist approach" (see Fig. 1). With the elaborationist approach, the definition of the application is built up progressively as you progress through from PIM to PSM to Code. When the PIM has created, the tool generates a skeleton or first-section PSM which the developer can then "elaborate" by adding more detail. Similarly, the final code is generated from PSM, and this can also be elaborated.

# 4 Column-Oriented NoSQL Database

There are four basic types of NoSQL databases: key-value, document-oriented, column-oriented and graph-oriented [2]. In this paper, we choose to focus on column-oriented NoSQL model. This model is considered to be the most efficient in terms of performance, for multi-criteria access queries (vertical data organization with columns-families).

The column-oriented databases were originally created by Facebook to store messages (non-instant) between users [17]. It is a key-value database extension, because the column model is more evolved, it is called super-column or column-family that a line identifier can store a structured set of data. A column-family has the following characteristics: the data is sorted, associated, and can contain an array of columns of unlimited size.

The storage of column-oriented databases is by column and not by row. These bases can evolve over time, either in number of rows or in number of columns. In other words, and unlike a relational database where columns are static and present for each row, the column-oriented databases are dynamic and present only when needed.

In column-oriented databases such as Cassandra [19] or HBase [20] there are some additional concepts that are the column-family, which are a logical grouping of rows. In the relational world this would be equivalent to a table. Cassandra offers an extension to the base model by adding an extra dimension called "Super Column" which itself contains other columns.

The concept of column-oriented databases is created by the big web actors, to meet the processing needs of large volumes of data precisely to manage large volumes of structured data. Often, these databases integrate a minimalist query system close to SQL called CQL [2].

In this paper, we choose the principle actor of column-oriented database such as Cassandra.

# 5 Source and Target Meta-Models

In our MDA approach, we opted for the modeling and template approaches to generate the column-oriented NoSQL database. As mentioned above, these approaches require a source meta-model and a target meta-model. We present in this section, the various meta-classes forming the UML class diagram source meta-model and the column-oriented NoSQL target meta-model.

## 5.1 UML source meta-model

Fig. 2 illustrates the simplified UML source meta-model based on packages including data types and classes. Those classes contain typed properties and they are characterized by multiplicities (lower and upper). The classes are composed of operations with typed parameters.

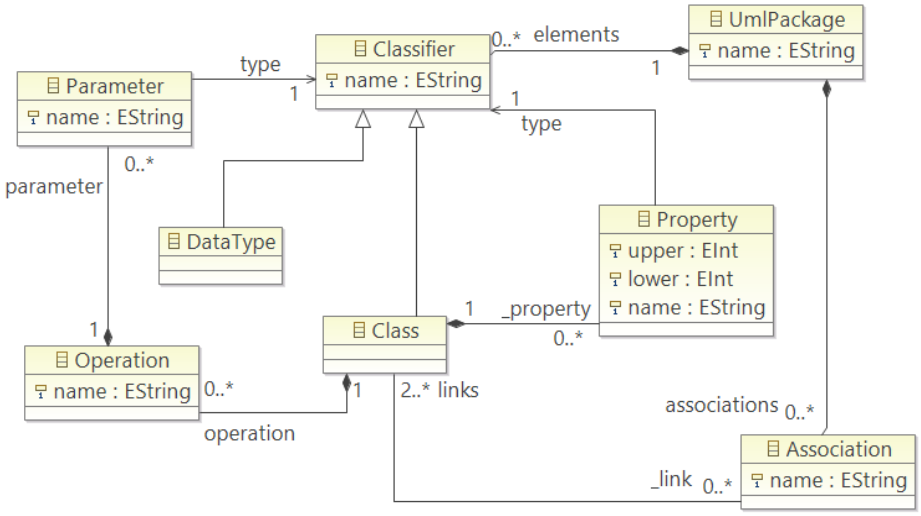


**Fig. 2.** Simplified UML source meta-model

- UmlPackage: is the concept of UML package. This meta-class is connected to the meta-class Classifier.
- Classifier: This is an abstract meta-class representing both the concept of UML class and the concept of data type.
- Class: is the concept of UML class.
- DataType: represents UML data type.
- Operation: is used to express the concept of operations of a UML class.
- Parameter: expresses the concept of parameters of an operation. These are of two types, Class or DataType. It explains the link between Parameter meta-class and Classifier meta-class.
- Property: expresses the concept of properties of a UML class. These properties are represented by the multiplicity and meta-attributes upper and lower.

The works [21, 22] contain more details related to this section topic.

### 5.2 Column-oriented target meta-model

To fully understand the data model used by Cassandra, it is important to define a number of concepts used:

- Keyspace: Appears as a namespace, this is usually the name given to the application.
- Column: Represents a value, its have three fields (see Fig. 3): its name, its value and a timestamp representing the date on which this value was inserted.

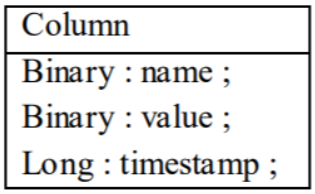


**Fig. 3.** Structure of the column element

- Super-Column: it's a list of columns (see Fig. 4), if you want to compare them with an SQL database, it's a row. It contains the key-value correspondence; the key identifies the super column while the value is the list of columns that compose it.

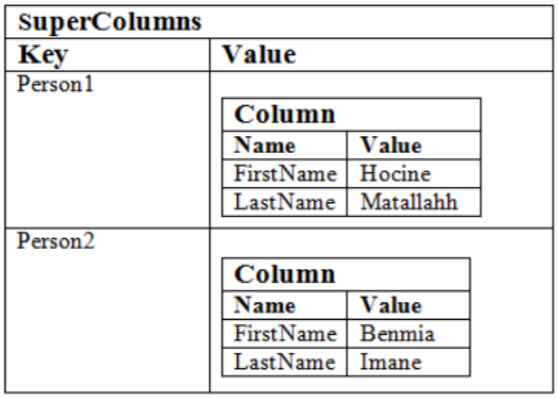


**Fig. 4.** Structure of the Super-column element

- Column-Family: it is a container of several columns or super-columns. Its notion is closer to the SQL table (see Fig. 5).

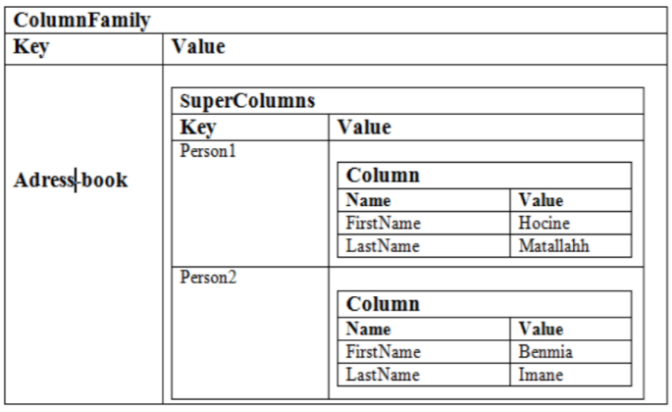


**Fig. 5.** Structure of the Column-Family element

Fig. 6 presents these concepts through the target meta-model.

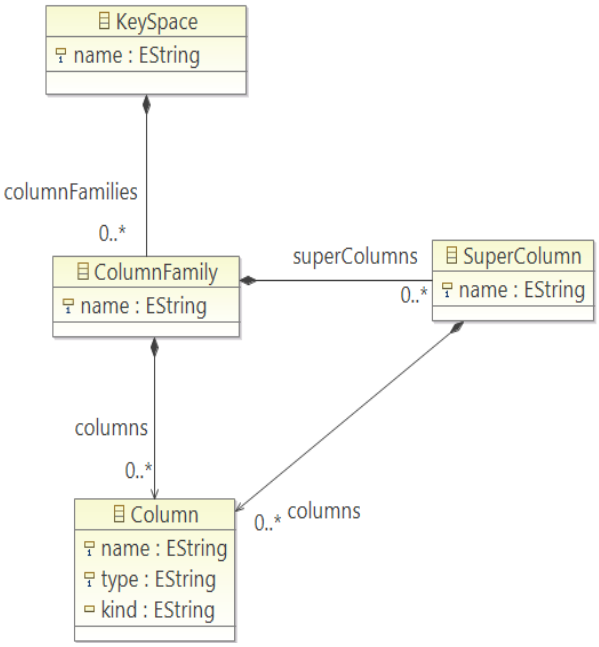


**Fig. 6.** Simplified column-oriented target meta-model

By default, we store the database in a single Keyspace. This Keyspace is comprised of a set of column-families. Each Column-family is identified by a unique identifier called "PrimaryKey" and contains several columns or super-columns that must be declared up front at schema creation time.

## 6 The Process of Transforming UML Source Model to Column-Oriented Target Code

We first developed ECORE models corresponding to our source and target meta-models. The development of many meta-models requires multiple model transformations. From these developed meta-models, M2M (Model to Model) and M2T (Model to Text) transformations are needed, to generate the code needed to create the column-oriented database.

We have implemented the M2M transformation algorithm (see section 6.1) using the QVT Operational Mappings language [23], and then the second M2T transformation is done with the Acceleo language [24] (see section 6.2).

### 6.1 The transformation rules M2M

This transformation uses, in entry, a model of the UML type, and in output a model of column-oriented database. The first transformation rule establishes the correspondence between all the elements of the UML package and the element of the Keyspace type of the column-oriented database. The purpose of the second rule is to transform each UML class and association into a family of columns by creating the columns and references for each column-family. It is a question of transforming each property of these classes in column, without forgetting to give names and types to the various columns.

Fig. 7 presents the principle part of the M2M transformation with QVT language.

UML2NoSQL.qvto

```
modeltype UML uses "http:///umlMM.ecore";
modeltype NoSQL uses "http://nosqlmm/1.0";
transformation UML2NoSQL(in umlModel: UML, out noSQLModel:NoSQL);
main() {
    umlModel.objects()[UmlPackage]->map UmlPackage2keySpace();
}
mapping UmlPackage::UmlPackage2keySpace () : KeySpace {
    name:= self.name;
    columnFamilies:=umlModel.objects()[Class] ->
            map classToColumnsFamily();
    columnFamilies+=umlModel.objects()[Association] ->
            map AssociationToColumnsFamily();
}
mapping Class::classToColumnsFamily():ColumnFamily{
name:=self.name;
columns:= object Column {
    name:= "id"+self.name;
    type:="uuid";
    kind:="PrimaryKey";
    };
columns+= self._property -> map propertyToColumn();
}
mapping Property::propertyToColumn():Column{
    name:=self.name;
    type:= self.type.name;
}
mapping Association::AssociationToColumnsFamily():ColumnFamily{
name:=self.name;
var cs : Sequence (Column);
self.links-> forEach(ls){
                cs+=object Column {
                    name:=ls.name+".Reference";
                    type:="uuid";
                    kind:="PrimaryKey";
                }
            };
columns:=cs;
}
```

**Fig. 7.** M2M transformation with QVT From UML to NoSQL model

### 6.2 The transformation rules M2T

The transformation M2T towards the creation code of column-oriented database in Cassandra is realized with Acceleo transformation language, and the writing of the transformation rules itself does not present any problems in practice. It simply boils down to creating a text file where the transformation rules are written.

Fig. 8 presents the transformation rules with Acceleo to generate a CQL file.

UML2NoSQL.mtl

```
[comment encoding = UTF-8 /]
[module UML2NoSQL('http://nosqlmm/1.0', 'http:///umlMM.ecore')]

[template public generateElement(aKeySpace : KeySpace)]
[comment @main/]
[file (aKeySpace.name.concat('.cql'), false, 'UTF-8')]
create KEYSPACE  [aKeySpace.name/] WITH REPLICATION =
{ 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

USE [aKeySpace.name/] ;

[for (cf:ColumnFamily|aKeySpace.columnFamilies)]
[ColumnFamilyToCQL(cf)/]
[/for]

[/file]
[/template]

[template public ColumnFamilyToCQL(cf: ColumnFamily)]
[let s:String='' ]
CREATE TABLE [cf.name/] (
[for (c:Column|cf.columns) separator(',')][c.name/] [c.type/] [/for],
PRIMARY KEY ([for (c:Column|cf.columns->
    select(f:Column|f.kind='PrimaryKey')) separator(',')]
        [c.name/]
[/for])
);
[/let]
[/template]
```

**Fig. 8.** M2T transformation with Acceleo to Generate a CQL code

## 7 Result

To validate our transformation rules, we conducted several tests.

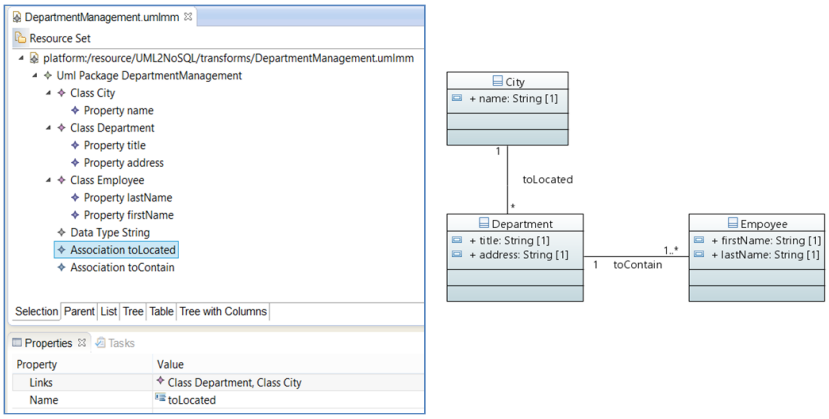


**Fig. 9.** UML source model: Class diagram EMF model and Class diagram instance model

For example, we considered the class diagram composed by the classes Department, Employee and City (see Fig. 9).

After applying the transformation on the UML source model, we generated the column-oriented PSM target model (see Fig. 10).

Fig. 10 shows the result after applying the transformation rules M2M.

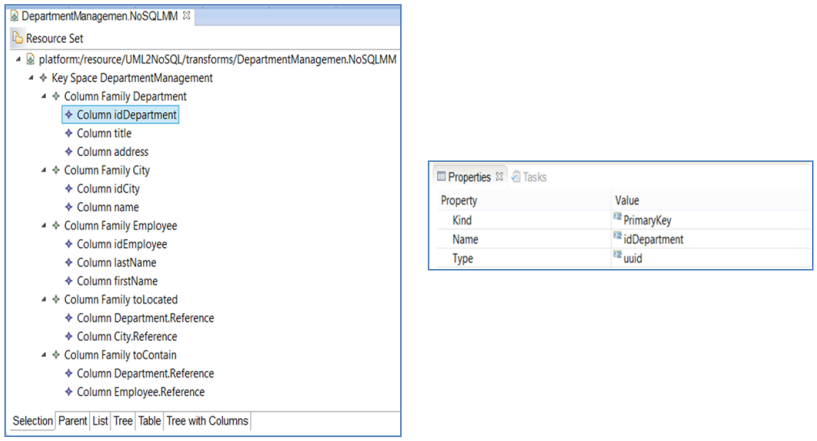


**Fig. 10.**Column-Oriented Cassandra PSM: Resource Set and their Properties

Fig. 11 illustrates the results of our M2T transformation. Our application generates a CQL code for a department management database on Cassandra platform.

DepartmentManagement.cql

```
CREATE KEYSPACE  DepartmentManagement WITH REPLICATION
= { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };

USE DepartmentManagement ;

CREATE TABLE Department (
idDepartment uuid ,
title text ,
address text ,
PRIMARY KEY (idDepartment)
);
CREATE TABLE City (
idCity uuid ,
name text ,
PRIMARY KEY (idCity)
);
CREATE TABLE Employee (
idEmployee uuid ,
lastName text ,
firstName text ,
PRIMARY KEY (idEmployee)
);
CREATE TABLE toLocated (
Department.Reference uuid ,
City.Reference uuid ,
PRIMARY KEY (Department.Reference,City.Reference)
);
CREATE TABLE toContain (
Department.Reference uuid ,
Employee.Reference uuid ,
PRIMARY KEY (Department.Reference,Employee.Reference)
);
```

**Fig. 11.**CQL file generated

## 8 Conclusion and Perspectives

In this paper, we have proposed an MDA approach to migrate UML class diagram representing a relational database to a column-oriented database. The transformations rules were developed using QVT to transform the class diagram into column-oriented model and then the automatic code generation using Acceleo with the goal to accelerate and makes easy the creation of NoSQL databases in Cassandra platform.

In the future, this work should be extended to allow the generation of other NoSQL Solutions such as document-oriented and graph-oriented. Afterward we can consider integrating other big data platforms like HBase, Redis, Neo4j and others.

## 9 References

[1] C. L. P. Chen, C. Zhang. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Inf. Sci., Vol. 275. pp. 314–347. https://doi.org/10.1016/j.ins.2014.01.015

[2] R. Cattell (2011). Scalable SQL and NoSQL data stores. ACM SIGMOD Record, Vol.39, No.4, pp.12–27. https://doi.org/10.1145/1978915.1978919

[3] D. Gwendal, S. Gerson, C. Jordi (2016). UMLtoGraphDB: Mapping Conceptual Schemas to Graph Databases. In: The 35th International Conference on Conceptual Modeling (ER).

[4] A. Abello (2015), Big Data Design. In: Proc. of the ACM Eighteenth International Workshop on Data Warehousing and OLAP, Australia. https://doi.org/10.1145/2811222.2811235

[5] Miller, J., Mukerji, J. (2003). MDA Guide Version 1.0.1, OMG, 2003.

[6] Abdelhedi Fatma, Ait Brahim Amal, Atigui Faten, Zurfluh Gilles (2017). MDA-based approach for NoSQL Databases Modelling, In: International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2017), Lyon, France, 28-31 August 2017. https://doi.org/10.1007/978-3-319-64283-3_7

[7] M. Chevalier, M. El Malki, A. Kopliku, O. Teste, R. Tournier (2015). Implementing Multidimensional Data Warehouses into NoSQL, In: International Conference on Enterprise Information Systems (ICEIS 2015), Barcelona, Spain, 2015. https://doi.org/10.5220/0005379801720183

[8] Yan Li, Ping Gu, Chao Zhang (2014). Transforming UML Class Diagrams into HBase Based on Meta-model. Information Science, Electronics and Electrical Engineering (ISEEE). https://doi.org/10.1109/infoseee.2014.6947760

[9] T. Vajk, P. Feher, K. Fekete, H. Charaf (2013). Denormalizing data into schema-free databases, In: 4th International Conference CogInfoCom. pp. 747–752. https://doi.org/10.1109/coginfocom.2013.6719198

[10] Abdelhedi, F., Ait Brahim, A., Atigui, F., Zurfluh, G. (2016). Big Data and Knowledge Management: How to implement conceptual models in NoSQL systems?, In: 8th International Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016), Porto, Portugal, 9 -11 November 2016. https://doi.org/10.5220/0006082302350240

[11] Abhinay B. Angadi, Akshata B. Angadi, Karuna C. Gull (2013). Growth of New Databases & Analysis of NOSQL Datastores. International Journal of Advanced Research in Computer Science and Software Engineering, Vol.3, Issue 6, June 2013.

[12] OMG, UML Infrastructure Final Adopted Specification, version 2.0, September 2003.

[13] OMG, Meta Object Facility (MOF), version 2.0, OMG, 2006.

[14] OMG, XML Metadata Interchange (XMI), version 2.1.1, OMG, 2007.
[15] Sara Gotti, Samir Mbarki (2019). IFVM Bridge: A Model Driven IFML Execution. International Journal of Online and Biomedical Engineering (iJOE). Vol. 15 No. 4. pp 111-126. https://doi.org/10.3991/ijoe.v15i04.9707
[16] P. Papajorgjin, P. M. Pardalos (2010).Towards a Model-Centric Approach for Developing Enterprise Information Systems. Enterprise Information Systems and Implementing It Infrastructures: Challenges and Issues. IGI Global; 1 edition. pp. 140-158. https://doi.org/10.4018/978-1-61520-625-4.ch010
[17] Radoslava S.K., Velin S.K., Nina S., Petia K., Nadejda B. (2019). Design and Analysis of a Relational Database for Behavioral Experiments Data Processing. International Journal of Online and Biomedical Engineering (iJOE). Vol 14, No 02 (2018). pp 117-132.
[18] D. Abadi, P. Boncz, S. Harizopoulos (2012).The Design and Implementation of Modern Column-Oriented Database Systems", Foundations and Trends in Databases: Vol. 5: No. 3, pp 197-280. https://doi.org/10.1561/1900000024
[19] Apache Cassandra, http://cassandra.apache.org/
[20] Apache HBase, https://hbase.apache.org/
[21] Oualid B., Saida F., Amine A., Mohamed B. (2018). Applying a Model Driven Architecture Approach: Transforming CIM to PIM Using UML. International Journal of Online and Biomedical Engineering (iJOE). Vol. 14, No. 9.pp 170-181. https://doi.org/10.3991/ijoe.v14i09.9137
[22] Karim Arrhioui, Samir Mbarki, Mohammed Erramdani (2018). Applying CIM-to-PIM Model Transformation for Development of Emotional Intelligence Tests Platform. International Journal of Online and Biomedical Engineering (iJOE). Vol. 14, No. 8. pp 160-168. https://doi.org/10.3991/ijoe.v14i08.8747
[23] OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.1, 2011.
[24] Acceleo, http://www.eclipse.org/acceleo

## 10 Authors

**Redouane Esbai** teaches the concept of Information System at Mohammed 1 University. He got his thesis of national doctorate in 2012. He got a degree of an engineer in Computer Sciences from the National School of Applied Sciences at Oujda. He received his M.Sc. degree in New Information and Communication Technologies from the faculty of sciences and Techniques at Sidi Mohamed Ben Abdellah University. His activities of research in the MASI Laboratory (Applied Mathematics and Information System) focusing on MDA (Model Driven Architecture) integrating new technologies XML, Spring, Struts, GWT, etc.

**Fouad Elotmani** is a PhD student from Mohammed First University (Nador, Morocco). His research interests at the MASI Laboratory (Applied Mathematics and Information System) include model driven modernization, Software development, modeling JEE Frameworks, and modeling architectures.

**Fatima Zahra Belkadi** PhD Student, she got her engineer Degree in software quality from the National School of Applied Sciences at Oujda. She is a researcher on studying the Big Data and their applications at MASI laboratory in Mohammed First University, Morocco.

Article submitted 2019-03-08. Resubmitted 2019-04-15. Final acceptance 2019-04-24. Final version published as submitted by the authors.