A Model-Driven Engineering: From Relational Database to Document-oriented Database in Big Data Context

Fatima Zahra Belkadi and Redouane Esbai
Department of Computing, Mohammed First University, University Complex, Post Box 658, Oujda, Morocco

Keywords: Big Data, MDA, Meta-Model, QVT, NoSQL, Document-oriented, Acceleo, JSON.

Abstract: In today's world multiple players in digital technology produce infinite amounts of data. Sensors, social networks or e-Commerce, they all generate information that is incremented in real time according to Gartner's 5V: in Volume, Velocity, Variety, Value, and Veracity. The digital transformation of companies leads an evolution of databases towards Big Data whose power has become increasingly strategic. The exploitation of the data is a key to a better understanding and management of the company and its markets. This obviously imposes an ability to generate data, to store it, to give it meaning and then to exploit it. At the same time, the modernization of the Big Data platform is essential to automate this process. In this paper, we explain how to design and apply transformation rules to transfer from an SQL relational database to a Big Data solution within NoSQL. For this, we use the Model Driven Architecture and the transformation languages as MOF 2.0 QVT (Meta-Object Facility 2.0 Query-View-Transformation) and Acceleo for describing the meta-models for the development of transformation model. The transformation rules defined in this work can generate, from the class diagram, a JSON files for creation document-oriented NoSQL database.

1 INTRODUCTION

Nowadays, artificial intelligence, mobiles, social media and internet of things are producing a huge amount of data whose size or type is beyond the ability of the traditional relational databases to capture, manage and process the data with low-latency, high volume, high velocity, and high variety. Big data bring many attractive opportunities to knowledge management (C. Fredriksson, November 2015).

Simultaneously, to extract knowledge from Big data, we have to face a lot of challenges, mainly related to Big data storage and process.(F. Abdelhedi, A. Ait Brahim, F. Atigui and G. Zurfluh, November, 2016) Using relational databases proves to be inadequate for all applications; particularly ones involving large volumes of data (A. Abello, 2015). As a result, a new type of databases has appeared, known as "NoSQL" data stores, that are able to handle Big data with high performance (A. Angadi, Ak. Angadi and Karuna. Gull, june 2013). On the other hand, a central problem facing the enterprise's developers is the building and storage of big data that are more complex than before.

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 (J.Miller and J.Mukerji, 2003).

Therefore, Abdelhedi et al. (F. Abdelhedi, A. Ait Brahim, F. Atigui, G. Zurfluh, November, 2016), generate respectively datasets of size 1GB, 10GB and 100GB. The experimental setup created; show how they can instantiate OLAP systems with column-oriented and document-oriented databases respectively with HBase and MongoDB. This method includes data transformation, data loading and aggregate computation. The entire process allows to compare the different approaches with each other.

This paper aims to rethink the work presented in (M. Chevalier, M. El Malki, A. Kopliku, O. Teste and R. Tournier. 2015). However, we propose an MDA approach for the development of NoSQL document – oriented databases from relational databases with implementation in MongoDB.

The rest of this article is ordered as follows. Section 2 represents a state of art (related Works). Section 3 states the objectives of this research paper and its background. Section 4 gives a formal representation of relational databases meta-model, the NoSQL Meta-model and the elaboration of rules' transformation that lead from the source model to the target model. The final section concludes this paper, and outlines future work.

2 RELATED WORKS

To the best of our knowledge, there are few solutions that have presented approaches for transforming the relational database into NoSQL systems.

Li and Vajk (Li, C., 2010), (Vajk T., Feher P., Fekete K., Charaf H., 2013), have investigated the process of transforming relational databases into a NoSQL model document –oriented databases.

Li (Daniel, G., Sunyé, G. and Cabot, J., 2016), have proposed an approach for transforming a relational database into HBase (column-oriented system).

Vajk et al. (Vajk T., Feher P., Fekete K. and Charaf H. 2013), defined a mapping from a relational model to document-oriented model using MongoDB.

Other studies have presented approaches to implement UML conceptual model into NoSQL systems.

Li et al. (Li Y., Gu P., Zhang C. 2014), propose a MDA-based process to transform UML class diagram into column-oriented model specific to HBase. Starting from the UML class diagram and HBase metamodels, authors have proposed mapping rules between the conceptual level and the physical one.

Gwendal et al. (Daniel, G., Sunyé, G. and Cabot, J. 2016) describe the mapping between a UML conceptual model and graph databases via an intermediate graph meta-model. These rules are specific to graph databases used as a framework for storing, managing and querying complex data with many connections. Generally, this kind of NoSQL databases is used in social networks where data are highly connected.

Chevalier et al (M. Chevalier, M. El Malki, A. Kopliku, O. Teste and R. Tournier. 2015) defined a set of rules to map a multidimensional model into two NoSQL models: column-oriented and document-oriented.

In other paper,of (F. Abdelhedi, A. Ait Brahim, F. Atigui, G. Zurfluh, November, 2016), they show how to store Big Data within NoSQL systems. For this, they use the Model Driven Architecture (MDA) that provides a framework for models' automatic transformation. Starting from a conceptual model that describes a set of complex objects, they propose transformation rules formalized with QVT to generate a column-oriented NoSQL model.

The main purpose of our work is to define a mapping from a relational model to NoSQL model: document-oriented, to formalize it and to automate it using QVT as a language of transformation. In Our solution we propose UML to NoSQL approach that offers the possibility to transform automatically a UML conceptual model into a NoSQL physical model. We define the transformations rules and we develop using the standard MOF 2.0 QVTand Acceleo aiming at the automated code generation with the goal to accelerate and build the creation of NoSQL databases in MongoDB platform.

3 MODEL DRIVEN ARCHITECTURE (MDA) APPROACH

To formalize and automate UML to NoSQL process, we use the Model Driven Architecture (MDA). One of the main aims of MDA is to separate the functional specification of a system from the details of its implementation in a specific platform (Hutchionson J., Rouncefield M. and Whittle J. 2014). This architecture defines a hierarchy of models from three points of view: Computation Independent Model (CIM), Platform Independent Model (PIM), and Platform Specific Model (PSM) (Bézivin J. and Gerbé O. 2001).

The MDA identifies several transformations during the development cycle (Sara Gotti, Samir Mbarki (2019). It is possible to make tree different type 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.

In this paper we chose two types of transformation, we start with the transformation PIM to PSM using the approach by modeling. This type of transformation will allow us to automatically generate a document-oriented NoSQL model from an RDBMS 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 JSON code for creating document-oriented NoSQL database.

4 DOCUMENT-ORIENTED NoSQL DATABASE

NoSQL databases draw upon several concepts and techniques to realize flexible data modeling and associated query languages, and horizontal scalability (Gudivada V. N., Rao D. and Raghavan, V. V., 2014). There are four types of NoSQL databases: Columnar Database, Graph Database, Key-Value Database and Document Database. In this paper, we choose to focus on Document databases.

The Document databases have the Document as a basic unity of data. Collections are groups of documents. The id field is reserved for the primary key.(Franca and Wilson Da Rocha, 2015)

Another important concept that will affect our decisions when designing a document is atomicity. (Franca and Wilson Da Rocha, 2015)

Now that we understood the method we can write our documents, let's take some examples of reality problems, such as how to write a data model that better describes the relationship between entities. (Franca and Wilson Da Rocha, 2015)

4.1 One to One

Figure1 illustrates the example that map patron and address relationships. To convert from the relational model to the oriented document model we must embed the address data in the patron data.

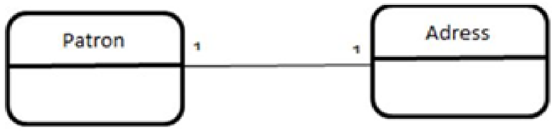


Figure 1: Simplified example mapping patron and address relationship.

4.2 One to Many

Figure 2 illustrates the example that maps publisher and books relationships. To avoid repetition of the publisher data in each document of books, we must use references and keep the publisher information in a separate collection from the book collection.

When using references, the growth of the relationships determines where to store the reference. If the number of books per publisher is small with limited augmentation, storing the book reference inside the publisher document may sometimes be useful. Otherwise, if the number of books per publisher is uncontrolled, this data model would lead to growing arrays.

To avoid growing arrays, stock the publisher reference inside the book document.

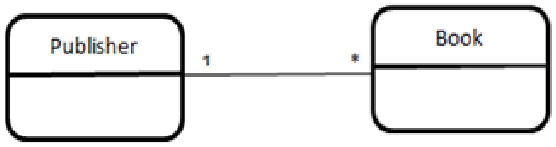


Figure 2: Simplified example mapping publisher and book relationship.

4.3 Many to Many

Figure 3 illustrates the example that maps user and group relationships.

A many-to-many relationship is not something trivial, even in a relational universe. In the relational world, this kind of relationship is often represented as a join table. While, in the non-relational one, it can be represented in many different ways.

The solution for this relationship is to store the reference in both documents.

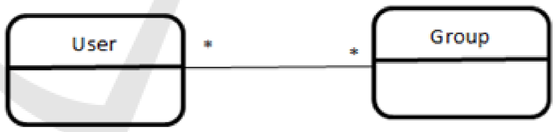


Figure 3: Simplified example mapping user and group relationship.

5 SOURCE AND TARGET META-MODELS

In our MDA approach, we opted for the modeling and template approach to generate the document-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 relational database source meta-model and the document-oriented NoSQL target meta-model.

5.1 Relational Database Source Meta-model

Figure 4 illustrates our source meta-model, this meta-model represents a version of the schemas of relational databases.

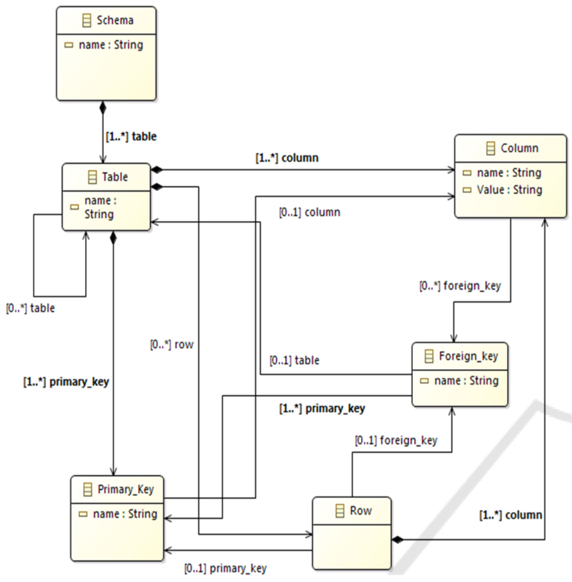


Figure 4: Relational database source meta-model.

We present here the different meta-classes to express the concept of tables of a relational database:

- **Schema**;
- **Table**;
- **Column**;
- **Row**;
- **Primary Key**;
- **Foreign Key**.

The work (A.Srai, F. Guerouate, N.Berbiche and H. Drissi, 2017) contains more details related to this section topic.

5.2 Document-oriented Target Meta-model

Figure 5 illustrates our target meta-model, this meta-model represents a version of the Document-Oriented databases.

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

- **Data_Base:** Appears as a namespace, this is usually the name given to the application;

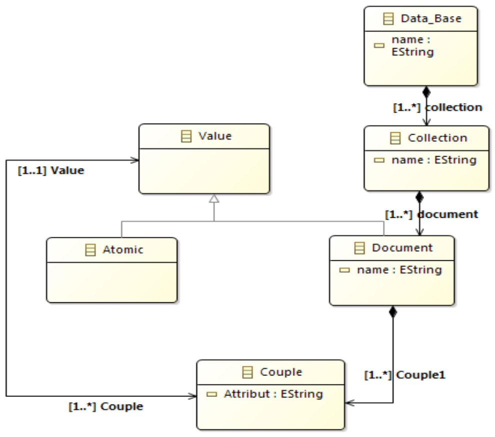


Figure 5: Document-Oriented target meta-model.

- **Document:** represent a hierarchy of elements which may be either atomic values or documents. In the NoSQL approach, the schema of documents is not recognized previously;
- **Collection:** is a group of documents;
- **Couple:** Document is defined by a set of couples, attribute = value
- **Value:** The value is atomic. It is itself composed by a nested document that is defined as e new set of couple (attribute, value). We can distinguish simple attributes whose values are atomic from compound attributes whose values are documents called nested documents.

6 THE PROCESS OF TRANSFORMING RDBMS MODEL TO DOCUMENT-ORIENTED TARGET CODE

We initial developed ECORE models corresponding to our source and target meta-models. The development of the many meta-models needs multiple model transformations. From these developed meta-models, M2M (Model to Model) and M2T (Model to Text) transformations are required, to generate the code required to create the document-oriented database. We have implemented the M2M transformation algorithm (refer to section 6.1) using the QVT Operational Mappings language (Franca, Wilson Da Rocha,2015), then the second M2T transformation is done with the Acceleo language (Acceleo) (refer to section 6.2).

Each paper must have at least one keyword. If more than one is specified, please use a comma as a separator. Keywords should appear justified, with a linespace exactly of 11-point, a hanging indent of 2-centimeters, spacing before of 48-point, no spacing after and font size of 9-point. The sentence must end with a period.

6.1 The Transformation Rules M2M

This transformation uses, within the entry, a model of the RDBMS type, and in output a model of the document-oriented database. The primary transformation rule establishes the correspondence between all parts of the schema of the relational database and therefore the element of the document-oriented database. The aim of the second rule is to transform each Table and his rows into a collection of documents by creating the documents and references for every collection. It's a matter of transforming every column of those tables in a couple of values, (Figure 6).

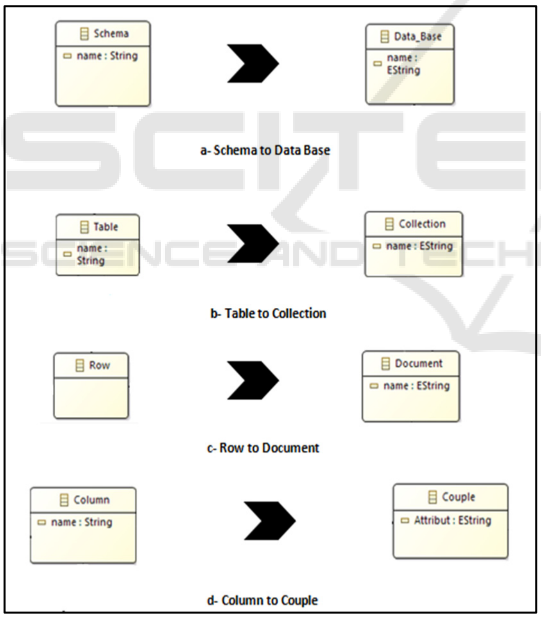


Figure 6: Rules of transformation from the RDBMS to NoSQL (Document).

The document is the basic unit of data. Collections are groups of documents. Making an analogy, a collection is similar to a table in a relational model and a document is a record in these tables. And finally, collections belong to a database.

Dissimilar to the relational model, where you must declare a table structure, a collection doesn't make obligatory a certain structure for a document. It is possible that a collection contains documents with completely different structures (Franca, Wilson Da Rocha,2015).

The principle part of the M2M transformation with QVT language (RDBMS2OD.qvto):

```
modeltype RDBMS uses
"http://www.example.org/bdr2nosql";
modeltype OD uses
"http://www.example.org/oDTarget";

transformation RDBMS2OD(in
RDBModel:RDBMS, out ODModel:OD);

main() {
   RDBModel.objects()[Schema]->map
schema2DataBase();
}

mapping Schema::schema2DataBase () :
Data_Base {
   name:= self.name;
   collection :=
RDBModel.objects()[Table] -> map
tableToCollection();

}

mapping
Table::tableToCollection():_Collecti
on{

name:=self.name;
document:= self.row -> map
rowToDocument();

}

mapping
Row::rowToDocument():Document{

name:=self.id;

self.columns ->forEach(c) {
   if c.foreign_key->size()=0 then {
         Couple1+= c.map
columnToCouple();
    }else {
         documents+=c.foreign_key.map
foreignKey2Document(c.value);
    }
    endif;

   }
}
mapping
Foreign_key::foreignKey2Document(c:S
tring):Document{
   name:=self.table.name;
   Couple1:=self.primary_key.column.m
ap fkToCouple(c)
}
```

```
mapping
Column::columnToCouple():Couple{
        Attribut:=self.name;
        valeur:=self.value;
}
mapping
Column::fkToCouple(c:String):Couple{
        Attribut:=self.name;
        valeur:=c;
}
```

6.2 The Transformation Rules M2T

The transformation M2T towards the creation code of the document-oriented database in MongoDB is complete with Acceleo transformation language, and also the writing of the transformation rules itself doesn't gift any issues in practice. It merely boils all the way down to creating a text file wherever the transformation rules are written.

The transformation rules with Acceleo to generate a JSON file (generate.mtl):

```
[comment encoding = UTF-8 /]
[module
generate('http://www.example.org/oDT
arget')]

[template public generateElement(aDB
: Data_Base)]
[comment @main/]
[for (c : Collection |
aDB.collection)]

   [file
(aDB.name.concat('/').concat(c.name)
.concat('.JSON'), false, 'UTF-8')]
        [collection2Document(c)/]
   [/file]
[/for]

[/template]
[template public
collection2Document(c:Collection)]

[for (D : Document | c.document)]
{
[for (cp : Couple |
D.Couple1)separator (',')]
"[cp.Attribut/]": "[cp.valeur/]"
[/for]
[if (D.documents->size()>0)],
[for (dc : Document |
D.documents)separator (',')]
"[dc.name/]": [ '[' /]
[document(dc)/] [ ']' /]
[/for]
[/if]
}
[/for]
[/template]
[template public
document(dc:Document)]
[for (cp : Couple | dc.Couple1)][for
(cp : Couple | dc.Couple1)separator
(',')]"[cp.valeur/]"[/for][/for]
[/template]
```

6.3 Results

For our transformation rules' validation, we have a tendency to conducted many tests

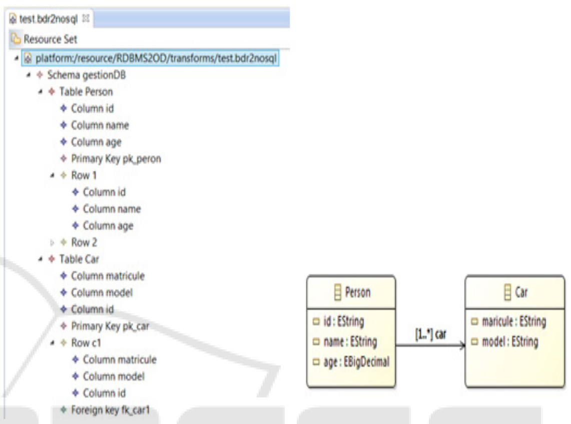

test.bdr2nosql
Resource Set
platform:/resource/RDBMS2OD/transforms/test.bdr2nosql
Schema gestionDB
Table Person
Column id
Column name
Column age
Primary Key pk_peron
Row 1
Column id
Column name
Column age
Row 2
Table Car
Column matricule
Column model
Column id
Primary Key pk_car
Row c1
Column matricule
Column model
Column id
Foreign key fk_car1

Figure 7: RDBMS source model: EMF model and instance model.

For example, we considered the schema composed by table Person and Car (see Figure.7).

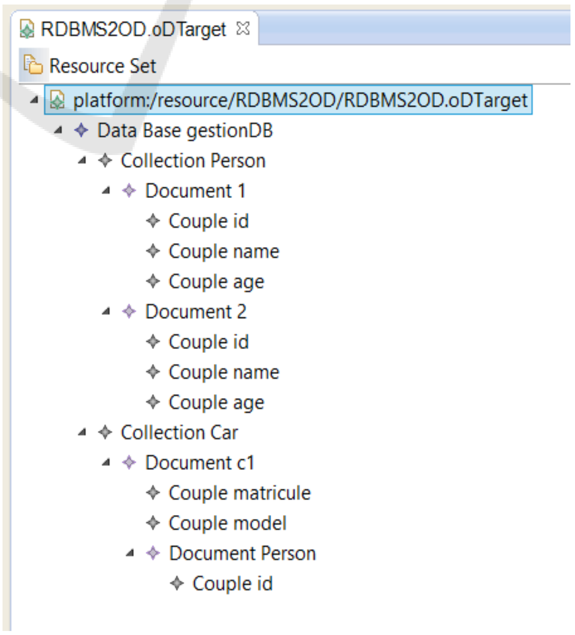
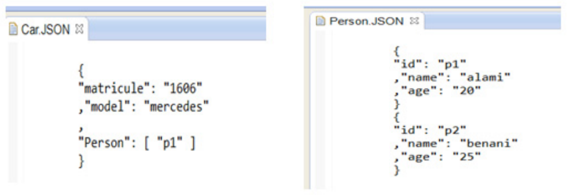

RDBMS2OD.oDTarget
Resource Set
platform:/resource/RDBMS2OD/RDBMS2OD.oDTarget
Data Base gestionDB
Collection Person
Document 1
Couple id
Couple name
Couple age
Document 2
Couple id
Couple name
Couple age
Collection Car
Document c1
Couple matricule
Couple model
Document Person
Couple id

Figure 8: Document-Oriented MongoDB PSM: Ressource Set and their Properties.

Figure 9 illustrates the results of our M2T transformation. Our application generates a JSON code for a GestionDB database on MongoDB platform.



```
{
"matricule": "1606"
,"model": "mercedes"
,
"Person": [ "p1" ]
}
```

```
{
"id": "p1"
,"name": "alami"
,"age": "20"
}
{
"id": "p2"
,"name": "benani"
,"age": "25"
}
```

Figure 9: The JSON Files generated.

7 CONCLUSIONS AND PERSPERTIVES

In this article, we've projected an associate degree of a MDA approach to migrate RDBMS model representing a relational database to a document-oriented database. The transformations rules were developed using QVT to transform the relational database into a document -oriented model and so the automated code generation using Acceleo with the goal to accelerate and build straightforward the creation of NoSQL databases in MongoDB platform. In the future, this work ought to be extended to permit the generation of different NoSQL Solutions like graph-oriented. Afterward, we are able to take into account integration different huge data platforms like HBase, Redis, Neo4j, and others. (Redouane Esbai, Fouad Elotmani and Fatima Zahra Belkadi, 2019).

REFERENCES

C. Fredriksson, (November 2015). "Knowledge Management with Big Data Creating New Possibilities for Organizations", in *The XXIVth Nordic Local Government Research Conference (NORKOM), pp.15-16.*

F. Abdelhedi, A. Ait Brahim, F. Atigui, G. Zurfluh, (November, 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, pp.235 -236.*

A. Abello, (2015). "Big Data Design: Processings of the ACM Eighteenth International Workshop on Data Warehousing and OLAP".

A. Angadi, Ak. Angadi, Karuna. Gull, (june 2013) "Growth of New Databases & Analysis of NOSQL Datastores". *International Journal of Advanced Research in Computer Science and Software Engineering, vol3, issue 6, p.1310.*

J. Miller, J. Mukerji, (2003) "MDA Guide Version 1.0.1", OMG, pp.1-3.

Li, C. (2010): Transforming relational database into HBase: A case study, In: *ICSESS, pp.683-686.*

Vajk T., Feher P., Fekete K., Charaf H. (2013), "Denormalizing data into schema-free databases", *CogInfoCom pp.748-756.*

Li Y., Gu P., Zhang C. (2014), "Transforming UML class diagrams into HBase based on meta-model.", In: *ISEEE, pp.720-724.*

Daniel, G., Sunyé, G., Cabot, J. (2016) : UMLtoGraphDB: Mapping conceptual schemas to graph databases. In: *Comyn-Wattiau, I., Tanka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, Springer, Cham, pp. 430-444.*

M. Chevalier, M. El Malki, A. Kopliku, O. Teste, R. Tournier. (2015): "Implementing Multidimensional Data Warehouses into NoSQL". *ICEIS, Barcelona, Spain.*

Hutchionson J., Rouncefield M., Whittle J. (2014): "Model-driven engineering practices in industry". In: *Science of Computer Programing 89 , pp. 153-158.*

Bézivin J., Gerbé O. (2001), "Towards a precise definition of the OMG/MDA framework". In: *ASE, pp.274-278.*

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.*

Gudivada V. N., Rao D., Raghavan, V. V. (2014), "NoSQL Systems for Big Data Management", In: *IEEE World Congress on Services, p.191.*

Franca, Wilson Da Rocha, packt publishing 2015: MongoDB Data Modeling, *ISBN : 978-1-78217-534-6, pp.21-32.*

A.Srai, F. Guerouate, N.Berbiche and H. Drissi (2017): An MDA approach for the development of data warehouses from Relational Databases Using ATL Transformation Language. In: *International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, pp. 3532-3538.*

OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.1, (2011).

Acceleo, [Online]. Available at: http://www.eclipse.org/acceleo.

Redouane Esbai, Fouad Elotmani and Fatima Zahra Belkadi, (2019): Model-Driven Transformations: Toward Automatic Generation of Column-Oriented NoSQL Databases in Big Data Context. *International Journal of Online and Biomedical Engineering (iJOE).– Vol. 15, No. 9. , pp.11-16.*

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.*

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 Biome-dical Engineering (iJOE). Vol. 14, No. 8. pp 160-168.*