International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 12 (2017) pp. 3532-3538

An MDA approach for the development of data warehouses from Relational Databases Using ATL Transformation Language

A. Srai[1], F. Guerouate[2], N. Berbiche[3] and H. Drissi[4]

System Analysis, Information Treatment and Integrated Management Laboratory

Superior School of Technologies of Sallee, Mohammadia School of engineering, Mohamed V University in Rabat, Avenue Prince Héritier -BP 227 Salé Médina – Maroc, Morocco.

[1]ORCID: 0000-0002-7942-4206

Abstract

These last years, the sources of data became very heterogeneous and massive. The term "big data" is born to indicate this phenomenon. Unfortunately, he becomes difficult to manage big data to power the decisional system, and also it increases the time of interrogation of data from data warehouses. Several research works focused recently, on the proposal for the architectures of data warehouses for this type of data ‘’big data’’, and on the implementation of new algorithms of interrogation of these warehouses to improve the time of the answers. This article proposes a Model-Driven Architecture (MDA) approach for the development of data warehouses independently of any execution platform, to allow the facilitation of the development of these data warehouses as well as the migration of information systems based on relational DBMS to systems NoSQL.

Keywords: Big Data; MDA approach; Data Warehouse; ATL.

INTRODUCTION

Nowadays, it is well recognized that model transformation is at the heart of model driven architecture (MDA) approaches and represents as a consequence one of the most important operation in MDA. The MDA (Model Driven Architecture) is a new discipline of software engineering that has emerged to assist software developers.

On the other hand, a central problem facing the decisional designers, is the building of data warehouses that are much more complex than before, they are complex because the volume of data is much larger. Therefore, to assist decisional designers in building data warehouse from relational data source, we propose an MDA approach for the development of data warehouses from relational databases with implementation in MongoDB or Hbase (NoSQL DBMS). In this approach, we benefited from the previous works especially the work of (Feki and Hachaichi, 2007; Jose-Norberto Mazón, Juan Trujillo,2007). The authors propose in different case studies, a DW design based on the use of the star schema and its different variations (snowflake and fact constellation) by using a relational approach, i.e. tables, columns, foreign keys, and so on. Although we consider this work as a fundamental reference. Our goal is to develop and implement a data warehouse by a series of transformations starting from the PIM (Platform Independent Model) to the PSM (Platform Specific Model), coded in ATL (ATL Transformation Language).

This paper is organized as follows: we begin in the first section with an introduction. The section 2 represents a state of art. Section 3 gives a formal representation of the UML meta-model, database meta-model and multidimensional schema meta-model. We elaborate also transformation rules, the transformation algorithm, the results of this transformation are presented in the same third section. The final section concludes this paper, and outlines future work.

STATE OF ART

Over the past 20 years, data has increased in a large scale in various fields. According to a report from International Data Corporation (IDC), in 2011, the overall created and copied data volume in the world was 1.8ZB (≈ 10^{21}B), which increased by nearly nine times within five years. This figure will double at least every other two years in the near future.

Under the explosive increase of global data, the term of big data [2,3] is mainly used to describe a massive volume of both structured and unstructured data that is so large it is difficult to process using traditional database and software techniques.

Big data is often characterized by 3Vs [4]: the great volume of data, the wide variety of data types and the velocity at which the data must be processed:

- Volume because the masses of data to be processed are constantly growing.
- Speed because collection, analysis and data exploitation must increasingly be in real time.

- Variety indicates the various types of data, which include semi-structured and unstructured data such as audio, video, webpage, and text, as well as traditional structured data.

Usually, Traditional data management and analysis systems are based on the relational database management system (RDBMS). However, such RDBMSs only apply to structured data, other than semi-structured or unstructured data. In addition, RDBMSs are increasingly utilizing more and more expensive hardware. It is apparently that the traditional RDBMSs could not handle the huge volume and heterogeneity of big data. The research community has proposed some solutions from different perspectives. For example, cloud computing is utilized to meet the requirements on infrastructure for big data, distributed file systems for solutions of permanent storage and management of large-scale datasets, and finally NoSQL[5,6] databases are good choices. In the next section we are more particularly interested, at first in NoSQL databases and secondly in datawarehouses.

A. NoSQL databases and Big Data

For over forty years, relational databases have been the leading model for data storage, retrieval and management. However, due to increasing volume of data, relational databases are becoming Inadequate. for this, NoSQL databases (non traditional relational databases) are becoming the core technology for of big data. The term NoSQL was first coined in 1988 to name a relational database that did not have a SQL (Structured Query Language) interface. It was then brought back in 2009 for naming an event which highlighted new non-relational databases, such as BigTable [8] and Dynamo, and has since been used without an "official" definition. Generally speaking, a NoSQL database is one that uses a different approach to data storage and access when compared with relational database management systems.

- Key-value Databases: Key-value Databases are constituted by a simple data model and data is stored corresponding to key-values. Every key is unique and customers may input queried values according to the keys. There are many representative key-value databases such as Dynamo, Voldemort[11] and Redis.

- Dynamo [7]: Dynamo is a highly available and expandable distributed key-value data storage system. It is used to store and manage the status of some core services, which can be realized with key access, in the Amazon e-Commerce Platform.
- Voldemort: Voldemort is also a key-value storage system, which was initially developed for and is still used by LinkedIn. Key words and values in Voldemort are composite objects constituted by tables and images. Voldemort interface includes three simple operations: reading, writing, and deletion, all of which are confirmed by key words.

- Column-oriented Database: The column-oriented databases store and process data according to columns other than rows. Both columns and rows are segmented in multiple nodes to realize expandability. The column-oriented databases are mainly inspired by Google's BigTable.

- Document Database: Compared with key-value storage, document storage can support more complex dataforms.

We will examine three important representatives of document storage systems, i.e., MongoDB [11], SimpleDB, and CouchDB.

- MongoDB: MongoDB is open-source and document-oriented database MongoDB is a collection of documents,Every document has an ID field as the primary key, query in MongoDB is expressed with syntax similar to JSON.
- SimpleDB: SimpleDB is a distributed databaseand is a web service of Amazon. Data in SimpleDB is organized into various domains in which data may be stored, acquired, and queried.
- CouchDB: Apache CouchDB [11] is a document-oriented database written in Erlang. Data in CouchDB is organized into documents consisting of fields named by keys/names and values, which are stored and accessed as JSON objects.

B. Data warehouses

Generally, the discussion around big data [9] focuses on, what we call, data warehouse, The data warehouse [18] (DWH), as defined by its inventor W.H. Inmon [10], is a collection of data which are subject-oriented, integrated, stamped, non-volatile, and used as a support of decision making. It is considered as a deposit of data that have been collected from heterogeneous and autonomous distributed sources. It is used for analytical tasks in business. Usually contains a very large amount of data. This is because of the scope of the period that the DWH must cover (historical data) and the diversity of data sources from which data are extracted.

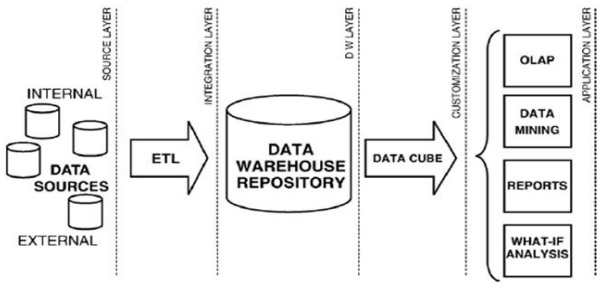


Figure 1. Architecture of data warehouse

C. Model Driven Architecture (MDA)

Model Driven Architecture (MDA) is an approach to application modeling and generation that has received a lot of attention in recent years. Championed by the Object Management Group (OMG), many organizations are now looking at the ideas of MDA [12,13,17] as a way to organize and manage their application solutions, this approach is founded on the creation of source models and transforming them to multiple levels of abstraction until having the source code automatically, the MDA includes the definition of several standards such as MOF [16], OCL [15], QVT [14], and XMI.

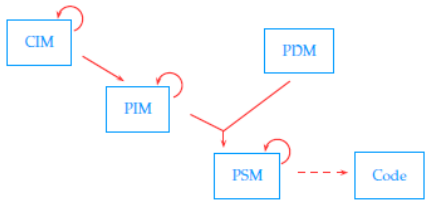


Figure 2. Transformation in MDA

In this paper, we present a solution based on Ecore Metamodel and ATL (Atlas Transformation Language).

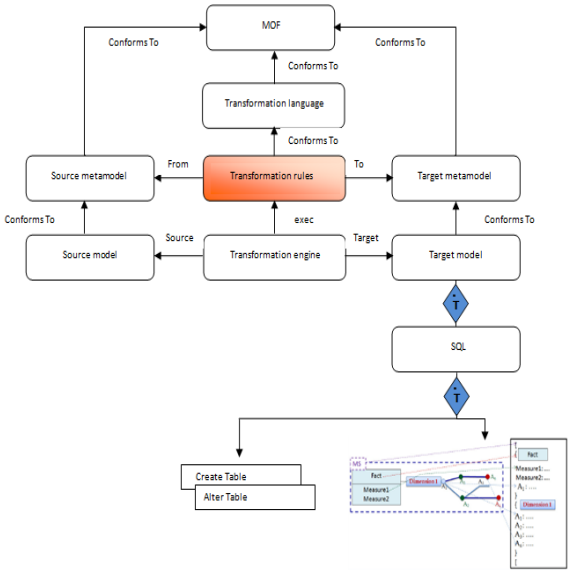


Figure 3. Simplified representaion of our approach

MDA has three modeling levels: *Model*, *Meta model* and *Meta Meta-model*. The Meta Meta-model is the highest

abstraction level, it defines the specification language of the Meta model, and its unique entities. The Meta model is the level where all instances of a Meta Metamodel are defined, its definition language permits the specification of models. The model level is the lowest one where instances of Meta-models are defined.

CONTRIBUTION OF OUR RESEARCH

Before developing our transformation algorithm between the source and target model, we present in figure 4, a graphic notation to express the mapping between the source and the target meta-model.

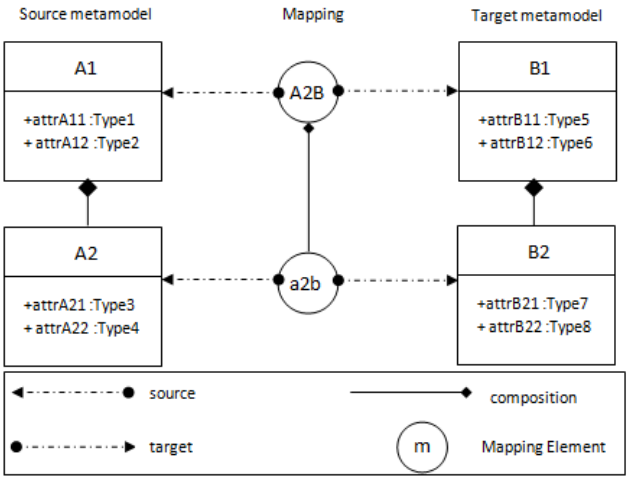


Figure 4. A proposed graphic notation to express mapping between source and target meta-model.

We present in this section, the various meta-classes forming the UML source meta-model and the relational database meta-model. The source meta-model represents a simplified version of UML model based on packages containing one or many classifiers, a classifier represents, in our case, a data type or a class. The classes are composed of operations with typed parameters, Figure 5 illustrates the source meta-model:

- ***Package***: represents the concept of UML package. This

meta-class is connected to the meta-class ***Classifier***.

- ***Classifier***: is an abstract meta-class representing the concept of UML class and the concept of data type.

- ***Class***: represents the concept of UML class.

- ***DataType***: represents UML data type.

- ***Operation***: is used to express the concept of operations of a UML class, each class can have one or more operations.

- Parameter: expresses the concept of parameters of an

operation. These are of two types, Class or DataType.

It explains the relation between Parameter meta-class and

Classifier meta-class.

- ***Property***: expresses the concept of properties of aUML class The association between ***Class*** meta-class and ***Property*** meta-class expresses the fact that a class is composed of properties.

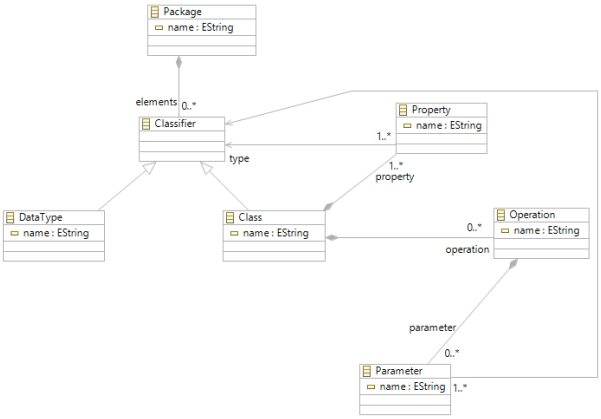


Figure 5. Simplified uml metamodel

We give in figure 6 the Meta model of the relational DB, This diagram is produced using EMF (Eclipse Modeling Framework) , we note that the most complex issue in MDA is the definition of transformations between a PIM [19] (Platform Independent Model) and a PSM (Platform Specific Model), for this, we have simplified the source meta model and the target meta model, we have identified the equivalent elements between them, and we have demonstrated this mapping by the proposed graphic notation in section 1. The implementation of this mapping rules is expressed with the ATL language , ATL is a model transformation language. we remember here that the implementation of mapping rules can be performed in three manners: by programming, by template or by modelling. We remember also the existing of QVT (Query/View/Transformation), is a standard set of languages for model transformation defined by the Object Management Group (consortium of more than 1000 enterprises). We are especially interested in applying the ATL language because it's the most widely used.

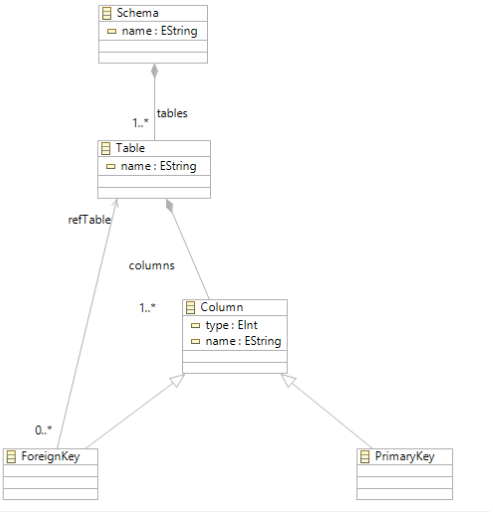


Figure 6. A simplified metamodel of a relational database

Figure 6 illustrates our target meta-model, This meta-model represents a version of the schemas of relational databases. We present here the different meta-classes to express the concept of tables of a relational database:

- ***Schema***: is the concept of schema in the relational databases. It contains a meta-attribute ***name,*** the metaclass

is connected by a meta-association to the metaclass ***Table.***

- ***Table***: represents the concept of table in the relational databases. It contains a meta-attribute ***name*** which represents the table name in the database. The metaclass is connected by a meta-association to the metaclass ***Column.***

- ***Column***: represents the concept of column in the database

tables in order to express the fact that column can be a primary key, or a foreign key, we decided to build a meta-class by the type of column.

- ***PrimaryKey:*** expresses a column as a primary key.

- ***ForeignKey***: expresses a column as a foreign key. The meta-class is connected by a meta-association to the meta-class ***Table*** to express the fact that the foreign key references the corresponding table.

A. The mapping rules

The aim of this section is to propose transformation rules to relational databases from source model, we mean model containing the various classes of our business model, the following list describes the various mapping rules :

*-**Rule1 :** each package in the UML source model is transformed to a schema in the target relational database model.*

*-**Rule2 :** each class in the UML source model is transformed to a table in the target relational database model.*

*- **Rule3 :** each property in the UML source model is transformed to a column in the target relational database model, where :*

- The name of the property is the name of the column.

- If the type of the property is a UML class, the column will be a foreign key in the relational database model.

- If the type of the property is a UML data-type, the column will have the adequate type in the target relational database model.

*- **Rule4 :** for each data type in uml packag , there are a data type in the relational schema.*

*- **Rule5 (**Fact**):** each source table in the relational database schema transforms into a fact if it contains at least one non-key numeric attribute.*

*- **Rule6 (**Measure**):** within each source table in the*

relational database schema identified as a fact, we transform all non-key numerical attributes into measures.

- Rule7 (*Dimension****):*** *within each source table in the relational database schema identified as a fact, we find all its foreign keys then, we transform each table referenced by a foreign key into a dimension for this fact.*

- Rule8 (*Parameter****):*** *each table transformed into a dimension provide parameters extracted from the fact table.*

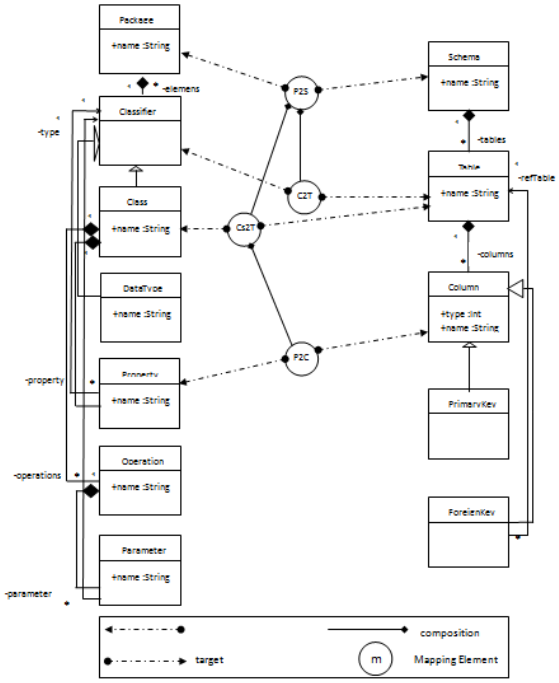


Figure 7. Mapping from UML to Database Meta-model

the mapping of two meta-models is presented using a graphical notation, this mapping is realized by a set of rules that specify the elements of the source and target meta-model.

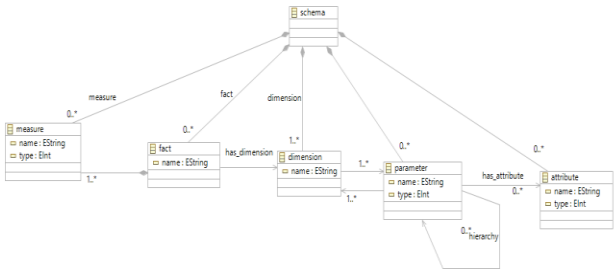


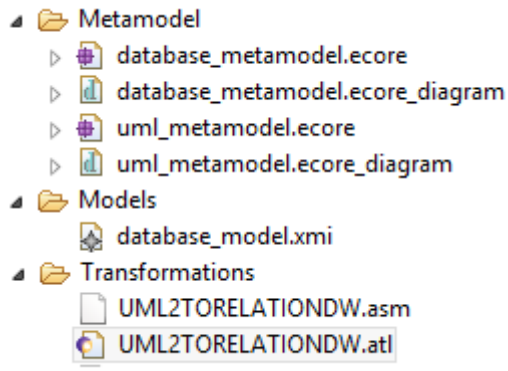
Figure 8. Multidimensional Meta model for datawarehouse

B. Algorithm

Let us remember here that our objective consists in applying the MDA approach in the development of DW, in fact, there are different kinds of model transformations are defined in MDA, in our case, we are interested in transformations (PIM to PIM) and (PIM to PSM), in the first case we have transformed a source UML model to a target UML model, in this paper we have presented only a few ATL transformation rules, the result is the refining module below :

```
-- @nsURI UML2=http://www.eclipse.org/uml2/3.0.0/UML

module UML2TOUML2;

create OUT : UML2 refining IN : UML2;

rule CLASS2CLASS {

        from c : UML2!Class -- class.allInstances()->
select(c|c.name=self.name)-> size()=1

        to  c1 : UML2!Class  (

                name <- c.name

        )

}

rule Property2Property {

        from p1 : UML2!Property -- context class inv :
self.property -> forall(p1,p2| p1<>p2 implies
p1.name<>p2.name)

        to p2 : UML2!Property (

                name <- p1.name,

                type <- p1.type,

                visibility <- #private --p1.visibility

        ),

        s : UML2!Operation(

                name <- 'set'+p1.name,

                class <- p1.class

        ),

        g : UML2!Operation(

                name <- 'get'+p1.name,

                class <- p1.class

        )

    }
```

Let us underline here that our fundamental objective is to produce a multidimensional DW [20,21] from a case study as illustrated in figure 9, for this, we have split the problem into two parts, the work presented in this paper and a future work that will present the case study, particularly to make the tasks easier. On the other hand, we use the following algorithm to generate the XML file:



Algorithm :

```
module UML2TORELATIONDW;
create OUT : RELATIONALDW from IN : UML2;
rule PACKAGETOSCHEMA {
        from    pck : UML2!Package
        to sch : RELATIONALDW!Schema (
                        name <- 'relational'+pck.name
                        -- *** (other bindings)
                ),
                 Tab : RELATIONALDW!Table(
                        name <- 'table'+'_'+pck.name
                        -- *** (other bindings)
                )
}
rule CLASSTOTABLE {
        from c : UML2!Class
        to t : RELATIONALDW!Table (
                        name <- c.name
                        -- *** (other bindings)
                ),
                 col : RELATIONALDW!Column(
                        name <-
'column_'+c.property.name
                        -- *** (other bindings)
                )

}
```

XML file :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:database="www.database.com">
 <database:Schema
name="relationalorg.eclipse.m2m.atl.engine.emfvm.lib.OclUndefined@592ac"/>
 <database:Table
name="table_org.eclipse.m2m.atl.engine.emfvm.lib.OclUndefined@592ac"/>
 <database:Table name="Client"/>
 <database:Column name="column_Client"/>
 <database:Table name="Product"/>
 <database:Column name="column_Product"/>
 <database:Table name="Sales"/>
 <database:Column name="column_Sales"/>
</xmi:XMI>
```

Our XML file represents, in fact, a transformation M2M model to model, in order to use this file with the Technology JET, to generate the SQL code.

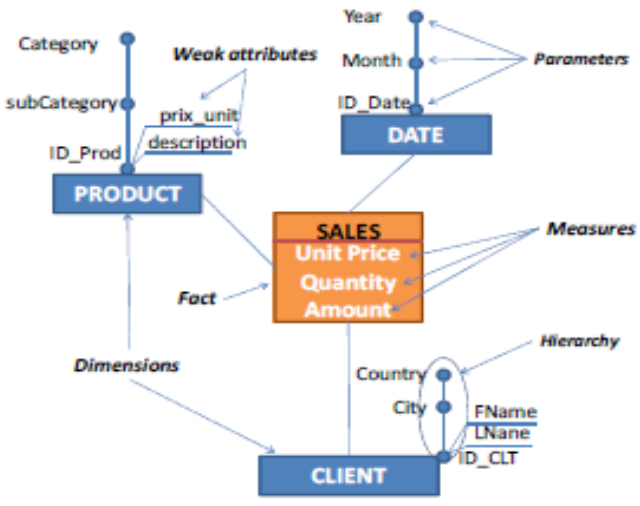


Figure 9. Simplified multidimensional schema

CONCLUSION AND FUTURE WORK

In this paper, firstly, we have defined the concept of big data and data warehouse. On the other hand, we have introduced our MDA approach for the development of datawarehouses by

a series of transformations starting from the PIM (Platform Independent Model) to the PSM (Platform Specific Model), coded in ATL (ATL Transformation Language). We have defined also, the transformation rules to transform a relational database model to a multidimensional model (star schema).

In perspective, we are also working on how to apply other execution platforms like PHP, MVC2 and .NET in the context of MDA approach.

REFERENCES

[1] Manyika J, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Byers AH (2016), "Big data: the next frontier for innovation, competition, and productivity," McKinsey Global Institute, 2011. http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_ innovation. Accessed 12 January 2016.

[2] Che D, Safran M, Peng Z (2013), "From big data to big data mining: challenges, issues, and opportunities," In: Lecture notes in computer science, vol 7827, pp 1–15.

[3] Michael K, Miller KW (2013), "Big data: new opportunities and new challenges," IEEE Comput 46(6):22–24.

[4] Wikipedia (2015), "Big Data," http://en.wikipedia.org/wiki/Big_data.

[5] Anwaar Ali, Junaid Qadir, Raihan ur Rasool, Arjuna Sathiaseelan, Andrej Zwitter and Jon Crowcroft, "Big data for development: applications and techniques," Big Data Analytics (2016) 1:2 DOI 10.1186/s41044-016-0002-4.

[6] Leavitt N, "Will nosql databases live up to their promise?," Computer. 2010;43(2):12–4.

[7] DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramanian S, Vosshall P, Vogels W, "Dynamo: amazon's highly available key-value store," In: ACM SIGOPS Operating Systems Review. ACM; 2007. p. 205–20.

[8] Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE, "Bigtable: A distributed storage system for structured data," ACM Trans Comput Syst (TOCS). 2008;26(2):4.

[9] Qin Yao, Yu Tian, Peng-Fei Li, Li-Li Tian, Yang-Ming Qian, Jing-Song Li, "Design and Development of a Medical Big Data Processing System Based on Hadoop," JMedSyst (2015) 39:23. DOI : 10.1007/s10916-015-0220-8.

[10] Inmon, W. H., "*Building the data warehouse*," Wiley, New York, 2005.

[11] Cattell R (2011), "Scalable SQL and NoSQL data stores," SIGMOD Rec 39(4):12–27.

[12] Kuznetsov, M.B., "MDA: New Concept of Software Integration," *Otkrytye sistemy*, 2003, no. 9, pp. 48–51.

[13] Kleppe, A., Warmer, J., and Bast, W., "*MDA Explained. The Model Driven Architecture: Practice and Promise*," Pearson Education, 2003.

[14] Request for Proposal: MOF 2.0 Query/Views/Transformations RFP, OMG document, April 2002, http://www.omg.org/cgi-bin/doc?ad/2002-04-10.

[15] Warmer, J. and Kleppe, A., "*The Object Constraint Language: Getting Your Models Ready for MDA*," Addison Wesley, 2003, 2nd Edition.

[16] *Meta Object Facility (MOF) 2.0 Core Specification*, OMG document, October 2003, http://www.omg.org/cgi-bin/doc?ptc/2004-10-15.