Journal of Theoretical and Applied Information Technology
15th October 2018. Vol.96. No 19
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# A NEW APPROACH FOR MIGRATION OF A RELATIONAL DATABASE INTO COLUMN-ORIENTED NOSQL DATABASE ON HADOOP

**[1]YOUNESS KHOURDIFI, [2]MOHAMED BAHAJ, [3]ALAE ELALAMI**

Department of Mathematics and Computer Science (Laboratory of Innovation for New Energy Technologies and Nano-Materials (LITEN))

Faculty of Sciences and Techniques, Hassan 1st University, Settat, Morocco

E-mail: [1]ykhourdifi@gmail.com, [2]mohamedbahaj@gmail.com, [3]elalamialae@gmail.com

## ABSTRACT

This article presents a new approach based on the "Object" concept, to successfully migrate a relational MySQL database to a column oriented HBase NoSQL database. The purpose of this article is to provide a new model of migration process divided into three phases, the first of which allows to obtain a copy of these metadata using the principle of semantic enrichment, and this to extract the different principles of the objects, including aggregation, inheritance and composition, the second phase of the process concerns the automatic generation of a New Optimised Data Model 'NODM' containing all relational database information in a flattened way. The last phase serves for the migration of the existing relational database into column-oriented database in the Hadoop ecosystem. The whole approach proposes a migration solution from a relational database to a NoSQL column-oriented database, which exploits the fast extraction of data columns for several types of applications, thus generating a better factor for analytic query performance, minimizes the input / output load of the disk, and reduces the amount of data being addressed from the disk.

**Keywords:** *Relational database, HBase, Data Migration, Semantic Enrichment, Schema Translation, BigData, Hadoop.*

## 1. INTRODUCTION

The digital world is developing very rapidly and becoming more complex in storage volume, variety of structures and velocity. This is mainly due to a change in our habits: what we expect from computers has changed; the democratization of smart phones and tablets, and the spread of social networks encourage the exchange and production of new content. The growth in the volume of data produced is increasing exponentially.

In addition to this massive production and exchange of data, there are data released by organizations and companies, known as Open Data: public transportation schedules, regional and government statistics, corporate network, store data.

Taking the site Planetoscope (http://www.planetoscope.com) as an example that estimates 74 460 billion e-mails were sent in the world in 2015 against 66 795 billion in 2013. In 2015, 204 billion e-mails were sent to the world (except spam) every day against 183 billion in 2013, or approximately 2 361 000 emails were received and sent per second. In 2015, there were 4.3 billion e-mail accounts open worldwide. A company of 100 people generates 13.6 T of CO2 each year if only by the use of e-mail, the equivalent of 14 Paris New York roundtrips by plane [1].

The impressive volume of data is closely linked with the frequent use of computers and with the dominance of giant web companies (Apple, Google, Amazon, and LinkedIn). They aim to encourage the ever-increasing use of their services. This increase in the consumption of their services mechanically translates into a growing demand for data processing and storage power that generates a rapid obsolescence of the IT architectures usually used: relational databases and application servers must give way to new solutions. The searches introduced a new database model called NoSQL databases that offers easy scalability of faster I/O operations and lower cost than traditional databases [2][3].

NoSQL databases don't follow any particular structure and don't support foreign keys, which facilitates data access [4]. The addressed problem is based on the inefficiency of relational databases that can no longer manage the growing demand for current applications [5]. Hence, the need to propose a new system that would be able to migrate the content of relational databases to NoSQL databases, while maintaining data integrity and keeping the application code intact [6]. In this perspective, we identify a significant number of proposed systems providing a solution to this problem; however, we regret the lack of relevant and optimized approaches in this direction. Our paper analyses recent approaches proposed in the literature to migrate relational databases to NoSQL databases [7][8], we mention these approaches because they don't benefit the advantages of NoSQL model, that manage to gather the tables in an optimized way, and proposes a new approach capable of automatically generating the column-oriented HBase scheme from the imported relational data from MySQL, using the object concept to enrich the relational database and benefit from the advantages of reducing the number of records in the target by exploiting those of NoSQL type.

The massive multiplication of the number of columns makes this model capable of storing one-to-many relationships, which makes it possible to cover many cases. On the other hand, simplistic queries are a constraint that will target column-oriented databases to applications that can be content with simplified data access in favor of increased performance, scalability or reliability.

In the Web world, column-oriented databases will support the progressive increase without sacrificing query-intensive functionality. Although they have been forged by the "big ones of the Web" these databases can also be adapted to other types of systems.

Some examples of storage use cases are:
•lists of articles for each user
•list of actions performed by a user
•event timeline maintained and accessed in real time
•mass data to analyze

Access to this type of information per page will be very fast because of the co-location of data provided by the organization of the columns sorted on the disk. Some users will be able to use this type of database to take advantage of their data distribution model and simply use it as key-value storage, while maintaining the possibility of richer data persistence in the future.

Our model is split into three parts, the first consists of the retrieval of data and useful information for establishing migration, the second part is represented in the form of a raw relational database and enriched by the different object concepts, the third part focuses on the migration of the relational database to the NoSQL HBase database using the different object concepts extracted during the phase of semantic enrichment.

This paper is organized as follows. Section II briefly reviews the HBase data model in the Hadoop ecosystem; Section III shows the related work on migrating SQL to NoSQL databases. Section IV describes the semantic enrichment that will play the role of the kernel of the approach, and Section V describes in detail our approach to automatically transforming the SQL to HBase database. Section VI presents the results of the experiment. Finally, a conclusion and some future directions will be explored.

## 2. BACKGROUND

### 2.1 From 3 V to 7 V

Big Data is a new technological field that is emerging to fling aside with the growing number of structured and unstructured data. It aims to offer an alternative to traditional database and analytics solutions. This technology allows a very fine analysis of massive data [9].

At the beginning, the given term **"BIG DATA"** was related to the 3V (**Volume, Variety, and Velocity**), the 3V are the dimensions of the big data:

**Volume:** Big Data is usually associated with this feature which describes the amount of data generated.

**Variety:** One of the components of Big Data is the diversity of data. They are called unstructured (or semi-structured) data because they are images, audio or video files, measurements. Generally, the variety refers to the number of data types.

**Velocity:** Velocity describes the frequency with which data is generated, captured, and shared, and refers to the speed of data processing.

Today, to enrich the 3 V's model has been added else 4 V's:

**Veracity:** presents truthfulness of the data.

**Value:** indicates useful outcomes.

**Variability:** refers to changes in data format, structure and semantics.

**Visualization:** describes presentation of big data.

7 V's of big data is presented in Fig. 1.

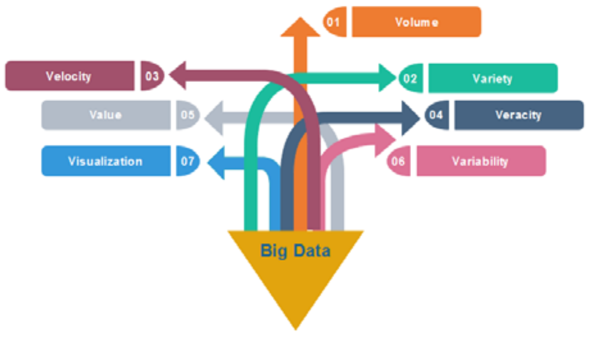

*Figure.1: The 7 V's of Big Data*

## 2.2 The Hadoop ecosystem :

### 2.2.2 Hadoop

Hadoop [10] is an acronym for the Distributed Object-Oriented Platform High Availability. Therefore, an open source platform offers storage and processing capabilities. It revolves around the two fundamental concepts: HDFS Hadoop Distributed File Systemand MapReduce, Figure 2 below Hadoop architecture.

Although it can work on a single machine, its true power is visible only in an environment composed of several computers. Hadoop is, therefore, the answer to a simple observation: the increase of the disk space does not go with the acceleration of the data reading. The solution would be to split the data into various parts to store them on multiple machines.

**What is HDFS?**

HDFS [11] is an acronym for Hadoop Distributed File System; this means that it uses the network to manipulate access to the system. Each component of the network can, therefore, access each resource of other computers that make up this network. In addition, HDFS is a system adapted to work with large volumes of data (1 GB and more). The great positive point of this system is universality.

**What is MapReduce?**

MapReduce [12] is a programming model that allows processing large volumes of data in several simultaneous processes. This model is based on two steps:

**Mapping (map tasks):** the developer defines a mapping function whose purpose is to analyze the raw data contained in the files stored on HDFS to output correctly formatted data.

**Reducing (reduce tasks):** this task retrieves the data constructed in the mapping step and analyses it in order to extract the most important information.

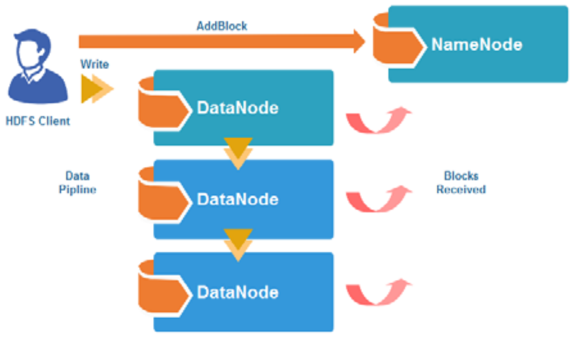

*Figure.2 : HDFS architecture*

### 2.2.3 HBase :

HBase is the Hadoop database [13][14][15], a distributed and scalable big data store. HBase has some features such as linear and modular scalability, strictly consistent reads and writes convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables. The HBase table in the database can store a series of rows. Each row is composed of three basic definitions: Row Key, Timestamp and Column. Row Key is a unique identity row in the table, sorted according to the dictionary order. The timestamp is the timestamp corresponding to each data operation, and it will record the different versions of the data. The column is defined as <Family> : <Qualifier>, which can specify the data type. The family identifies a list of clusters, and the number is constant when setting up the table. While Qualifier belongs to the list of family clusters in the column, column clusters can be any number of columns. The columns in the cluster columns can be dynamically added and does not require predefined. The data in the HBase have no types, which are stored in a binary format. The HBase index is the row, column, the column name and timestamp. Through Via Row Key and <Family> : <Qualifier> can fully determine a data unit value in tables. Figure 3 below HBase client ecosystem.

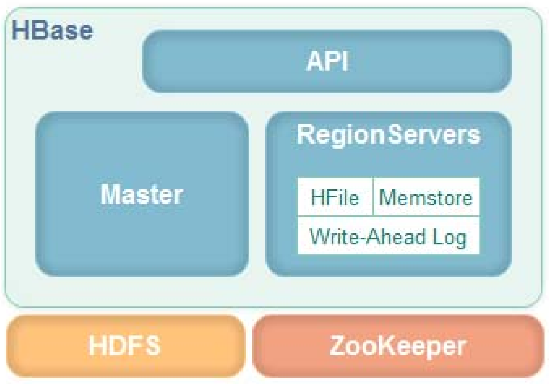

*Figure.3 : HBase architecture*

## 3. RELATED WORK

In the relational database migration engineering that follows a rigid structure for structuring data collection from various applications, there is significant research to solve the problem of building relational databases already in production in applications that work on the NoSQL paradigm [16] and provide flexibility in structuring data, for easier access to data. Among these studies, there are certain migration hypotheses with a single target; other researchers chose to work with an intermediary to migrate a relational database to multiple models.

Abdelhedi F discussed the process of transforming a conceptual schema, represented by a DCL and UML into a column-oriented NoSQL physical schema [17]. It distinguishes by tree layer: Computation Independent Model, Platform Independent Model, and Platform Specific Model. The Computation Independent Model coincides to the user specifications that, commonly, are indicated in natural language. The Platform Independent Model is the conceptual model. It could be avoided into different families of NoSQL databases.

Liu C [18] proposed a database migration method RDBMS relational database management systems to HBase and demonstrates the realization of this semi-automatic migration design method in the data migration process, Chevalier M [19] investigated the implementation of multidimensional data warehouses based on column-oriented NoSQL systems, and defined a set of rules for mapping star schemas in two NoSQL: column-based and document-based models. The experimental part is carried out using the TPC referential reference system. Their experiments show that rules can actually instantiate such systems (star schema and network). They also analyze the differences between the two NoSQL systems considered. In their experiments, HBase (column-oriented) is faster than MongoDB (document-oriented) in terms of loading time.

El Alami A [20] presented some approaches and methods of migration in database engineering, and offered an approach to transform a relational database to a document-oriented NoSQL database Mongo DB. The approach is based on a number of rules to allow promoting the determined structure, to exploit the constraints defined on the schema. It’s based on foreign key to establish connection between two types of DBMS to be able in order to carry out a dialogue. The whole migration is concentrated on a bridge that will act as a gateway that will connect the two DBMS.

Ghotiya S, et al. [21] provided a literature review of some of the recent approaches proposed by various researchers to migrate data from relational databases to NoSQL databases.

Lee, C.H. [22] proposed a mechanism for an automatic transformation of SQL schemas to NoSQL from MySQL to HBase database. The proposed method follows NoSQLs DDI principles: Denormalization, Duplication and Intelligent keys. In this approach, related tables are aggregated into a big table and then most suitable key is selected, which is called row key, to identify each row, it also use the traditional Web-based content management system (CMS) to sidestep the cross-table query in the NoSQL database,

Li C, et al. [23] proposed a new approach that transforms a relational database into HBase, includes two phases, the transformation into HBase schema, and the second is the experimentation of a set of nested schemas for relations between two schemas. They establish a comprehensive review touching the abilities of different databases according to burden detecting data created by performance management tools. They conclude that the Cassandra assessments continuous latency time during read/write processes. As for HBase demands less write latency but gives huge read latency time. Therefore, a superior capacity for handling distributed large input files are shown in Hbase rather than the Cassandra database.

Li Y, et al. [24] proposed a method based on the ideas of Model Driven Architecture (MDA) that transforms UML Unified Modeling Language class diagrams into HBase based on the Meta-model.Their method is reclaimable after the model mapping is built and the schema for HBase could be generated directly. The major inconvenient of their method is the source model's meta-model they created is not totally compatible with the conceptual standard, which means class diagrams and the conceptual data model which are not obey with this meta-model need to be remodeled before any transformation. Something else is the definition of the model mapping is comparatively fundamentally.

Rocha L, et al. [25] proposed a NoSQL Layer that migrate Relational databases to MongoDB databases without having to change the application code. The proposed approach keeps the entire structure of the source database and injects data as NoSQL model. SQL requests are converted to NoSQL requests performing them into NoSQL database.

Serrano D, et al. [26] suggested a method for transforming data schemas for the RDBMS to HBase, the method consists of a set of HBase-organization guidelines and a four-step data transformation process.

Zhao G, et al. [27] proposed a new HBase schema to migrate from the relational database to HBase, which supports multiple nesting.They apply a graph transforming algorithm to involve all required content.

Column-oriented databases are organized into column families. This type of grouping is similar to the concept of a table in a relational database. Although they are organized into tables, their layout is totally different. Thus while the columns of a relational database are static and present for each row, those of a column-oriented database are dynamic and present only for the lines concerned. In other words, it is possible to add columns to a row at any time and the cost of storing a null is 0. It's not just about storing the fields of an entity, but also one-to-many relationships. The possible queries are simple. It is possible to make requests by key, or set of keys and to add a predicate on a range of columns. The query system is minimalist and it greatly simplifies the design of these databases, in favor of performance. It is then possible to make queries that concern a name or a range of names of super-columns thus making it possible to obtain in return a list of super-columns and all their contents.

Our approach takes an existing relational database as input and produces an HBase database output with the use of metadata and a set of treatment. We use the principle of semantic enrichment to extract different characteristics of the object concept from the physical schema of the RDB, including aggregation, inheritance and composition. We optimize the model proposed by El Alami et al. [28] to make it more efficient and more lightweight, which eliminates some unnecessary element in our migration approach, and which is essential in their migration approach from a RDB to ORDB.

## 4. SEMANTIC ENRICHMENT

Semantic enrichment is a process of analyzing and examining a database to capture its structure and definitions at a higher level of meaning. This is done by improving the representation of the structure of an existing database in order to make its hidden semantics explicit.

We mean enriching the content/context of the data by labeling, categorizing and/or classifying the data in relation to each other, with dictionaries and/or other basic reference sources. In its simplest form, this means adding additional contextual information to some existing data.

In the semantic enrichment phase, we use the passed model to preserve the semantic database, create the NoSQL schema, and inject the data with the transformation algorithms.

Definition of the New Optimized Data Model (NODM)

The NODM model is a rich flattened relational model, divided into four entities, defined as follows:

**NODM := {C| C := <Cn, Classification, Attribute (An,tag), Contributor>}**

With Cn is the table name, Classification to define the relationship between tables by the object concept. For Attribute there is An as the attribute name, and tag to distinguish between PK and FK keys and a simple attribute. Finally, Contributor to list tables in relation with Cn.

NoSQL engines solve the problem of NULLs, but there are two more things to report. First, NULLs are a real problem in the relational model because they hinder queries by their three-state logic. In this case, problem solved. As far as space utilization of NULLs is concerned, it is minimal in RDBMS, and you cannot really save storage space on NoSQL engines. The lack of relationship between the tables generally leads to a lot of data redundancy and so, NULL or not NULL, NoSQL engines occupy much more storage space than a properly standardized model RDBMS. This is more of a gain in clarity for programming than storage.

## 5. OUR APPROACH TO AUTOMATIC TRANSFORMATION FROM SQL SCHEMA TO HBASE

The migration to a column-oriented database is necessary in several cases because the data is added on one dimension which is technically simpler and faster: the data are concatenated one after the other, thus support much higher write speeds with very low latency. A better scalability since the development of data is done only on one dimension their partitioning is simpler to perform and can be distributed across multiple servers.

### 5.1 Transformation rules

For the migration from a relational database to a NoSQL database, a set of rules has been established to ensure the integrity constraints to obtain an HBase database.

**Simple Class and aggregation:** For the transformation of a simple or aggregated class, we will create a table with the name of the relational table having the same name as the old one in the RDB by adding a name of ColumnFamily, is fn_TableName and for the attributes, they are represented by columns qualify.

**Class inheritance and association:** For the migration of a class inherited or associated, one proceeds to the creation of the two types, the mother classes then the classes girls, for the first classification one will create a table in the Row Key with the primary key of the main table, and for the second classification we will integrate it as familiar columns.

**Composition class:** The composition class is integrated into the class that collaborates with it according to Contributor = C1.Cn and C1.Classification = Composition, the class becomes a familiar column; the creation of the versions is established according to the number of records that appears in the relational database by a specific selection method.

For the detection of different object concept, we proceed by the inheritance detection via a data dictionary and a processing set, which is based on the comparison between primary keys to remedy problems related to non-standardized database.

Association detection is based on the detection of composite keys and the absence of primary keys, which are the migration results of primary keys when moving from the conceptual model to the logical model during the conception of the database.

Aggregation is a particular association which schematizes two classes ("aggregated") and ("aggregate") named respectively by C1 (E1, E2) and C2 (E3, E4) such that C2 is shareable and independent of C1. The semantic aspect is more important than the conceptual aspect. The aggregation relation is subjective whereas the association relation is pejorative leaving a doubt about the belonging of the class. The association is less precise and semantically weak, but remains correct.

The detection of composition is done when a class ("component") is not shareable, and dependent on a single class ("composite"). The dependency between the two classes prevents the referenced object from being deleted as long as the dependency is linked to the physical schema.

All transformation phases are shown in Fig.4.

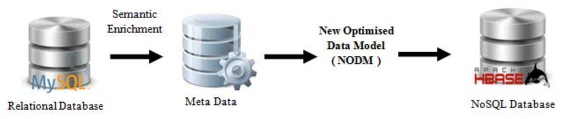

*Figure.4 : Transformation Overview*

### 5.2 Transformation algorithm :

To get the interaction of the HBase database, we have three methods, including a dedicated shell by running the HBase shell command, a native JAVA API that can inject it into a JAVA program, and finally external APIs like REST or Thrift.

In our approach, we use a Java API to create an HBase database from a relational MySQL database. The approach is focused on a three-layer step, the first layer will be dedicated to data recovery and other useful information to establish migration, the second layer will play the role of the core application that will be a flattened relational database form enriched by the different object concepts, the third layer will start the migration from the relational database to the HBase target database via the metadata captured in the first layer and the different object concepts captured during the semantic enrichment phase.

The whole application is based on two methods, the first method is a global selection and the second method is a specific selection in order to obtain the number of versions for the composition class and to obtain a personalized selection according to the class identifier that collaborates with the composition.

**// Algorithm producing HBase table**

```
Foreach Class C NODM do
tableName= resultset.getObject(tablename)
if C.Classification = Simple || C.Classification = Agregation then
    // Instantiating table descriptor class with table name
HTableDescriptor(TableName.valueOf(tableName));
    // Adding column families to table descriptor
tableDescriptor.addFamily(new HColumnDescriptor(”fn_”+tableName));
   // Execute the table through admin
admin.createTable(tableDescriptor);
String t[][] = db.selectAll(tableName);
for (int ii = 0; ii <t.length; ii++) //we start from 0 to get the name of attribute
for (int jj = 0; jj< t[ii].length; jj++) {
if C.Attribut.tag=pk then
Put p = new Put(Bytes.toBytes("t[ii][jj]"));
end if
p.add(Bytes.toBytes(“fn_”+tableName),
Bytes.toBytes(t[0][0]),Bytes.toBytes(t[ii][jj]));
end for
end for
else if C.Classification = inherBy then
   // Instantiating table descriptor class with table name
tableName= resultset.getObject(tablename)
HTableDescriptor(TableName.valueOf(tableName));
 // Adding column families to table descriptor
tableDescriptor.addFamily(new HColumnDescriptor(”fn_”+tableName));
for (String v: C.Contibutor.getValues())
req = C.Contibutor.getNomTable();
for(inti = 0; ii <table.length; i++)
ifC.Cn=req&& ((C.Classification= Inherts)|| (C.Classification= Association)|| (C.Classification= contributor)) then
tableDescriptor.addFamily(new HColumnDescriptor(”fn_”+req));
end if
end for
end for
    }
    // Execute the table through admin
admin.createTable(tableDescriptor);
String t[][] = db.selectAll(tableName);
Key=null;
Val=null;
for (inthh = 0; hh<t.length; hh++)
 //we start from 0 to get the name of attribute
for (intff = 0; f < t[hh].length; ff++) {
ifC.Attribut.tag=pk then
            Val = t[hh][ff];
            Key = t[0][0] ;
end if
end for
end for
for (int ii = 0; ii <t.length; ii++)
 //we start from 0 to get the name of attribute
for (int jj = 0; jj< t[ii].length; jj++) {
if C.Attribut.tag=pk then
              Put p = new Put(Bytes.toBytes("t[ii][jj]"));
end if
p.add(Bytes.toBytes(“fn_”+tableName), Bytes.toBytes(t[0][0]),Bytes.toBytes(t[ii][jj]));
end for
end for
for (String v: C.Contibutor.getValues())
req = C.Contibutor.getNomTable();
for(inti = 0; ii <table.length; i++)
if C.Cn=req&& ((C.Classification= Inherts)|| (C.Classification= Association)) then
                   String t[][] = db.selectAll(req);
for (int ii = 0; ii <t.length; ii++) //we start from 0 to get the name of attribute
for (intjj = 0; jj< t[ii].length; jj++) {
if C.Attribut.tag = pk then
 Put p = new Put(Bytes.toBytes("t[ii][jj]"));
end if
p.add(Bytes.toBytes(“fn_”+req), Bytes.toBytes(t[0][0]),Bytes.toBytes(t[ii][jj]));
end for
end for
```

```
else if C.Cn=req&&C.Classification=
composition)then
String t[][] = db.selectAllByChampO(req , Key,
Val)
 q=fn_+req;
q.setMaxVersions(t.length);
for (int iii = 0; iii <t.length; iii++)
 //we start from 0 to get the name of attribute
for (intjjj = 0; jj< t[iii].length; jjj++) {
ifC.Attribut.tag=pk  then
Put p = new Put(Bytes.toBytes("t[iii][jjj]"));
end if
p.add(Bytes.toBytes("fn_"+req),
Bytes.toBytes(t[0][0]),Bytes.toBytes(t[iii][jjj]));
end for
end for
end if
end for
// Saving the put Instance to the HTable.
hTable.put(p);
System.out.println("data inserted");
// closing HTable
hTable.close();
end for
```

This algorithm traverses the generated model set from the relational database, and for each classification check an instance of the description table is created by the name extracted from the NODM's C.CN database, then we add the column families from the description table instantiated before, then via the simple selection method we establish a selection of the names of the attributes via the table of the RDB (we start with the value 0 in our loop to capture the name of the attributes. The incrementation of the variable will give rise to the selection Data). For the versions where there will be a nesting of data we will use the specific method of selection to capture the number of versions and to make a selection via the identifier of the table which enters in collaboration with the aggregated table.

**// The global selection method**
**Definition of the selectAll method**

```
public String[][] selectAll(String tableName) {
String req = "SELECT * FROM " +tableName;
                try {
Statement sql = db.createStatement();
ResultSetrs = sql.executeQuery(req);
ResultSetMetaDatarsm = rs.getMetaData();
            int columns = rsm.getColumnCount();
            String data[][];
            rs.last();
            int rows = rs.getRow() + 1;
            data = new String[rows][columns];
for (int I = 1; I <=columns; i++) {
            data[0][i-1] = rsm.getColumnName(i);
                                    }
                                    int row = 1;
                                    rs.beforeFirst();
                                    while (rs.next()) {
            for (inti=1; i<=columns; i++) {
            data[row][i-1] = rs.getString(i);
                                                }
                                    row++;
                                    }
                                    return data;
                        }
catch (Exception e) {
                                    e.printStackTrace();
                                    returnnull;
                        }
            }
```

**// The specific selection method**
**Definition of the selectAllByChamp method: specific request**

```
public String[][] selectAllByChampO(String
tableName,Stringchamp,String name) {
                        String req = "SELECT * FROM "
+ tableName +" where "+ champ +"='"+name+"'";
                                    try {
int type =
ResultSet.TYPE_SCROLL_INSENSITIVE;
int mode = ResultSet.CONCUR_UPDATABLE;
                                    Statement
sql=db.createStatement(type,mode);
ResultSetrs = sql.executeQuery(req);
ResultSetMetaDatarsm = rs.getMetaData();
                                    int columns =
rsm.getColumnCount();
                        System.out.println(columns);
                                    String data[][];
                                    try {
                                                rs.last();
                                    } catch (Exception e) {
            System.err.println("Erreur sur rs.last()");
                                    }
            int rows = rs.getRow() + 1;
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
data = new String[rows][columns];
for (inti = 1; i<=columns; i++) {
data[0][i-1] = rsm.getColumnName(i);
                    }
                    int row = 1;
                    rs.beforeFirst();
                    while (rs.next()) {
for (inti=1; i<=columns; i++) {
data[row][i-1] = rs.getString(i);
                              }
                              row++;
                    }
                    return data;
          }
          catch (Exception e) {
                    e.printStackTrace();
                    returnnull;
          }
}
```

## 6. EXPERIMENTAL RESULT

In our model, new concepts have been exploited to build the transformation of relational databases into column-oriented HBase databases, making it an efficient model with relevant results.

For a column-oriented database, the notion of column does not really make sense. The data management model is based on a notion of couple {key, value}. The name of the column can be seen as the key. There is indeed no notion of column as it can be heard in a relational database. So to avoid any ambiguity with the notion of column for relational databases, we have reasoning in terms of Map, exploiting the object concept and the relations between objects that we extracted as it appears in table 1. Thus, our base relational data has become a column-oriented database that we can define as a Map-oriented database. The data in a Map-oriented database (column) is identified by a unique key (equivalent to a primary key for relational databases). Thus, we associate with these identifiers a set of data composed of a key pair, value (hence the notion of Map). In the relational world, these keys are called columns. The columns of a column-oriented database are dynamic and present only for the rows concerned.

To demonstrate the effectiveness and validity of our approach, we used a relational database, presented in Fig. 5, we have also developed a prototype, realizing these algorithms.

Below is the entire relational database that acts as a source database, from which we will proceed to the migration.

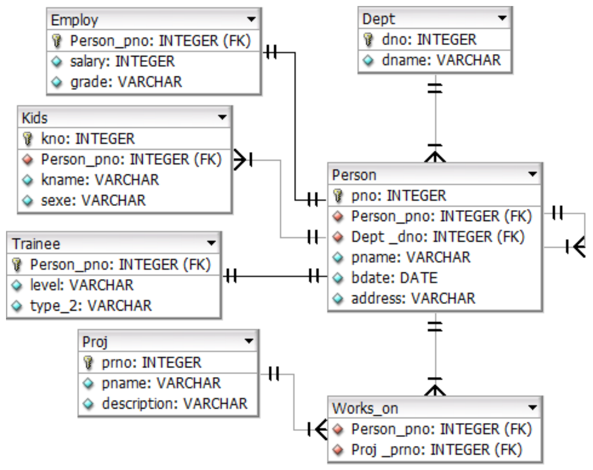

*Figure.5 : Relational databases*

Table 1 shows the results of the flattened relational database enriched by the different concepts of objects.

The results were evaluated, and represented in Fig.6. The result is an optimal source HBase database that belongs to the column-oriented NoSQL family, which encapsulates several tables in a single table according to our migration approach to exploit the concept of column-oriented databases.

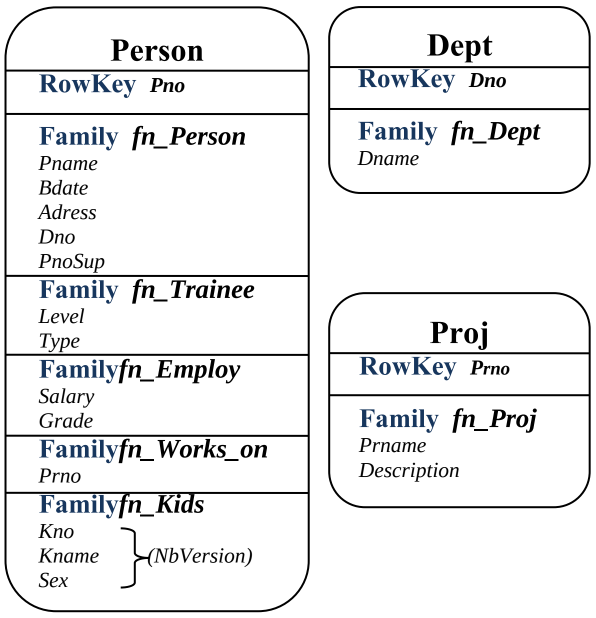

*Figure.6 : HBase schema after migration*

*Table 1. : Results of NODM generation*

| Cn | Classification | Attribut | | Contributor |
|---|---|---|---|---|
| | | An | tag | |
| Person | InherBy | Pno | PK | Kids |
| | | | | Works_ on |
| | | | | Trainee |
| | | | | Employ |
| | | Pname | | |
| | | Bdate | | |
| | | Adress | | |
| | | Dno | FK | Dept |
| | | PnoSup | FK | Person |
| Trainee | Inherts | Pno | FK | Person |
| | | Level | | |
| | | Type | | |
| Employ | Inherts | Pno | FK | Person |
| | | Salary | | |
| | | Grade | | |
| Works _on | Association | Prno | FK | Proj |
| | | Pno | FK | Person |
| Dept | Simple | Dno | PK | Person |
| | | Dname | | |
| Proj | Simple | Prno | PK | Work_o n |
| | | Prname | | |
| | | Description | | |
| Kids | Compositio n | Kno | PK | |
| | | Kname | | |
| | | Sex | | |
| | | Pno | FK | Person |

## 7. CONCLUSION

This paper provides a comprehensive solution to the problem of RDB migration to HBase.

Our model uses an existing MySQL relational database as input and produces aHBase column-oriented database as output. We use the principle of semantic enrichment to extract the different features of objects, including aggregation, inheritance and composition that are represented in a New Optimized Data Model (NODM).

This model is used to create a new way to store in a column oriented database which is focalized on Map structure. The purpose of this article is not to expose the details of the limitations of traditional (relational) database management systems, but to present a database system for large data management that can use a dynamic column. All the migration is done in an automatic manner, a prototype is made which prove the effectiveness of the approach.

The first phase of the migration process is the retrieval of data and information for establishing migration, the second phase deals with the automatic generation of a New Optimized Data Model that contains all the information related to the relational database, and finally, the last phase that inject the relational data to the HBase column-oriented database in the Hadoop ecosystem.The migration approach is used when there is a need to write heavy applications and used whenever we need to provide fast random access to available data.

## REFERENCES

[1] https://www.planetoscope.com/Internet-/1024-emails-envoyes-dans-le-monde.html.

[2] P. Atzeni, F. Bugiotti, L. Cabibbo, and R. Torlone, “Data modeling in the NoSQL world,”*Comput. Stand. Interfaces, 2016.*

[3] R. Yangui, A. Nabli, and F. Gargouri, “Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study,” *Procedia Comput. Sci., vol. 96, no. September, pp. 255–264, 2016.*

[4] D. Liang, Y. Lin, and G. Ding, “Mid-model design used in model transition and data migration between relational databases and NoSQL databases,” *Proc. - 2015 IEEE Int. Conf. Smart City, SmartCity 2015, Held Jointly with 8th IEEE Int. Conf. Soc. Comput. Networking, Soc. 2015, 5th IEEE Int. Conf. Sustain. Comput. Commun. Sustain. 2015, 2015 Int. Conf. Big Data Intell. Comput. DataCom 2015, 5th Int. Symp. Cloud Serv. Comput. SC2 2015, pp. 866–869, 2015.*

[5] C. H. Lee and Y. L. Zheng, “SQL-To-NoSQL Schema Denormalization and Migration: A Study on Content Management Systems,” *in Proceedings - 2015 IEEE International*

*Conference on Systems, Man, and Cybernetics, SMC 2015, 2016.*

[6] Y. T. Liao, J. Zhou, C. H. Lu, S. C. Chen, C. H. Hsu, W. Chen, M. F. Jiang, and Y. C. Chung, “Data adapter for querying and transformation between SQL and NoSQL database,” *Futur. Gener. Comput. Syst., 2016.*

[7] L. Stanescu, M. Brezovan, and D. D. Burdescu, “Automatic Mapping of MySQL Databases to NoSQL MongoDB,” *vol. 8, pp. 837–840, 2016.*

[8] J. Xu, M. Shi, C. Chen, Z. Zhang, J. Fu, & C. H. Liu, (2016, August). ZQL: A unified middleware bridging both relational and NoSQL databases. *In Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C (pp. 730-737). IEEE.*

[9] P. Zikopoulos, & C. Eaton,"Understanding big data: Analytics for enterprise class hadoop and streaming data". *McGraw-Hill Osborne Media 2011.*

[10]G. Turkingeton, Hadoopbeginner’s guide, 2013.

[11]T. White, Hadoop in practice, 2009.

[12]S. Guo, Hadoop operations and cluster management cookbook, 2013.

[13]L. George, HBase: The Definitive Guide, O'Reilly Media, Inc, USA (2011).

[14]T. White, Hadoop: The Definitive Guide, O'Reilly Media, Inc, USA, 3rd Revised edition (2012).

[15]C.X. Li, Transforming relational database into HBase: A case study, Software Engineering and Service Sciences (2010), 683–687.

[16]Atzeni, P., Bugiotti, F., Cabibbo, L., and Torlone, R. (2016). Data modelling in the NoSQL world. Computer Standards and Interfaces.

[17]F. Abdelhedi, A. A. Brahim, F. Atigui, and G. Zurfluh, “Big Data and Knowledge Management: How to implement conceptual models in NoSQL systems?”*KMIS. 2016.*

[18]L., Chen, F., Zhicheng, Y., Zhengqiu, et al. General Research on Database Migration from RDBMS to HBase. *In : 2015 International Symposium on Computers&Informatics. French: Atlantis Press. 2015. p. 124-237.*

[19]M. CHEVALIER, M. EL MALKI, A. KOPLIKU, et al. “Implementation of multidimensional databases in column-oriented NoSQL systems”. *In : East European Conference on Advances in Databases and Information Systems. Springer, Cham, 2015. p. 79-91.*

[20]A. EL ALAMI, and M. BAHAJ. “Migration of a relational databases to NoSQL: The way forward”. *In : Multimedia Computing and Systems (ICMCS), 2016 5th International Conference on. IEEE, 2016. p. 18-23.*

[21]S. Ghotiya, J. Mandal, and S. Kandasamy, “Migration from relational to NoSQL database,” *in IOP Conference Series: Materials Science and Engineering, 2017.*

[22]C. H. Lee and Y. L. Zheng, “Automatic SQL-to-NoSQL schema transformation over the MySQL and HBase databases,” *in 2015 IEEE International Conference on Consumer Electronics - Taiwan, ICCE-TW 2015, 2015.*

[23]L. Chongxin. Transforming relational database into HBase: A case study. *In : Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on. IEEE, 2010. p. 683-687.*

[24]Y. Li, P. Gu, and C. Zhang, “Transforming UML class diagrams into HBase based on meta-model,” *in Proceedings - 2014 International Conference on Information Science, Electronics and Electrical Engineering, ISEEE 2014, 2014.*

[25]L . Rocha, F. Vale, E. Cirilo, D. Barbosa, & F. Mourão, " A Framework for Migrating Relational Datasets to NoSQL".*In Procedia Computer Science of International Conference On Computational Science, Volume 51, 2015, Pages 2593–2602.*

[26]D. SERRANO, D. HAN, et STROULIA, Eleni. From relations to multi-dimensional maps: towards an SQL-to-HBase transformation methodology. *In : Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on. IEEE, 2015. p. 81-89.*

[27]G. Zhao, L. Li, Z. Li, and Q. Lin, “Multiple nested schema of HBase for migration from SQL,” *in Proceedings - 2014 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2014, 2014.*

[28]M. Bahaj, and A. Elalami. "The migration of data from a relational database (RDB) to an object relational (ORDB) database." *Journal of Theoreticaland Applied Information Technology 58.2 (2013).*