World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:12, 2015

Data Migration Methodology from Relational to NoSQL Databases

Mohamed Hanine, Abdesadik Bendarag, Omar Boutkhoum

***Abstract*—**Currently, the field of data migration is very topical. As the number of applications developed rapidly, the ever-increasing volume of data collected has driven the architectural migration from Relational Database Management System (RDBMS) to NoSQL (Not Only SQL) database. This very recent technology is important enough in the field of database management. The main aim of this paper is to present a methodology for data migration from RDBMS to NoSQL database. To illustrate this methodology, we implement a software prototype using MySQL as a RDBMS and MongoDB as a NoSQL database. Although this is a hard engineering work, our results show that the proposed methodology can successfully accomplish the goal of this study.

***Keywords*—**Data Migration, MySQL, RDBMS, NoSQL, MongoDB.

I. Introduction

CURRENTLY, there are many types of databases in which large amount of data are stored. Sometimes, it is necessary to migrate data from one type of database to another or to create new database implemented in another one and move data from old database to a new one. In these cases, the process of data migration consists of three steps called ETL (Extract, Transform and Load) [1]: (1) Extraction data from the source database, (2) Transformation data, and (3) Migration of data to the target database.

There is a variety of reasons for data migration, including server, storage equipment replacements or upgrades. In this regard, we only state few of these reasons [1]-[3]:

- Upgrading to new version database: In the case of upgrading of software equipment in the organization, including new version of database, we need to migrate data to new database.
- Existing database is insufficient: In the case of a large increase of stored data, there is insufficient capacity or speed of database.
- Changing of organization policy: In the case of changing security or another type of policy in organization, we need to upgrade to better database.
- Economic problems in organization: This can lead to uninstalling commercial and expensive database software and installing open-source and cheaper database instead.
- Unifying various applications and databases: We need to unify various databases to one database specific to ensure better efficiency and consistency of data.

In this article, two types of databases will be used for data migration. Source database is relational, and target database is a recent one in the market: NoSQL database [4].

- Relational database: A relational database or RDBMS is a database based on a relational model that was developed by Edgar Codd in 1970 [5]. The basic notion of this database is the separation into tables for storing all these data. The tables are structured in rows and columns, where they can be linked with each other by foreign keys. A big advantage of this database type is its ease to use, so the untrained business users can create easy own databases. However, the huge growth of new applications that depend on storing and processing big amount of data which added more challenges to the RDBMS, and where the classical SQL systems being inappropriate in a variety of ways, lead to a new database model called NoSQL.
- NoSQL Database: NoSQL databases have emerged tremendously in the last years owing to their less constrained structure, scalable schema design, and faster access compared to relational databases. The main feature that makes difference in the model of NoSQL data is that it does not use the table as storage structure of the data. In addition, its schema is very efficient in handling the unstructured data. The NoSQL database takes many modeling techniques like key-value stores, document data model, and graph. The following illustrates this clearly [6], [7]:

- Key-value stores: Data is stored as values with a unique key assigned to each value. Also, this type of NoSQL database allows for keeping high performance in reading and/or writing. Currently, the best solutions having adopted the system of key-value are Voldemort, Redis and Riak.
- Document stores: The concept of this database which is based on documents is a kind of extension of the key/value database in which the value is represented as a document containing data represented in standard formats (JSON: JavaScript Object Notation, XML, etc.). All documents are stored in collections (equivalent to tables in SQL). The advantage of database oriented documents is to retrieve a set of hierarchically structured information from a unique key. The most current implementations are

H.M. is with the Department of Computer Science, Laboratory of Engineering and Information Systems, Faculty of Sciences Semlalia, Cadi Ayyad University, Marrakesh, Morocco (corresponding author to provide phone: +212 677 34 63 41; e-mail: m.hanine@uca.ma).

B.A. is with the Department of Mathematics and Computer Science, Faculty of Polydisciplinary, Safi, Cadi Ayyad University, Morocco (e-mail: A.bendarag@uca.ma)

B.O. is with the Department of Computer Science, Laboratory of Engineering and Information Systems, Faculty of Sciences Semlalia, Cadi Ayyad University, Marrakesh, Morocco (e-mail: o.boutkhoum@uca.ma).

CouchDB and MongoDB.

- Graph databases: An important aspect of the database-oriented graph is the use of index. This means that each element contains a pointer to its adjacent element and does not require indexing of every element. The most current implementations are Neo4j, HypergraphDB and FlockDB.

This paper presents a methodology for data migration from relational database to NoSQL. During data migration, the NoSQL schema will be created automatically. The proposed methodology will be verified on data migration from MySQl relational database to NoSQL database implemented in MongoDB.

The main reasons that lead us to choose MongoDB as target database are the fact that it has many advantages:

- Open source.
- Document oriented NoSQL databases.
- Functioning as a distributed centralized architecture, it replicates data on multiple servers with the master-slave principle, allowing a greater fault tolerance.
- The number and type of fields in a document are not defined previously.
- It is usable with major development languages (C, Java, PHP, Python ...) via drivers.
- Availability in multiple environments such as Linux and Windows
- A very powerful base in terms of speed.
- Characterized by dynamic schema, high scalability, and optimal query performance.

Fig. 1 shows the model structure of MongoDB, and Table I shows some SQL concepts and terminology and their correspondences in MongoDB [8], [9].

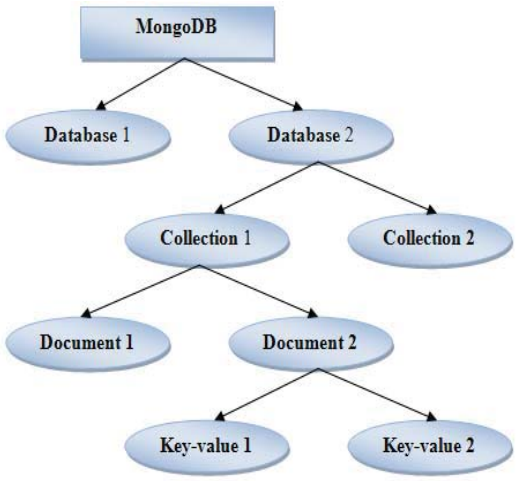

Fig. 1 Model structure of MongoDB

The rest of the paper is organized as follows. The second section explains concisely the proposed methodology. In third section, the software prototype for demonstrating the proposed methodology is illustrated. Finally, conclusions and further research are offered in the last section.

TABLE I
BASIC SQL VS MONGODB SYNTAX

SQL Terminology/ Concepts/Syntax	MongoDB Terminology/ Concepts/Syntax
table	collection
row	document or bson document
column	field
table joins	embedded documents and linking
primary key	primary key
create table users (id mediumint not null auto_increment, user_id varchar (30), age number, status char (1), primary key (id))	db.users.insert({ user_id: "abc123", age: 55, status: "a"})
alter table users add xdate datetime	db.users.update({ },{ $set: { xdate: new date() } },{multi: true})
alter table users drop column xdate	db.users.update({ },{ $unset: { xdate: "" } },{multi: true })
create index idx_user_id_asc on users(user_id)	db.users.ensureindex({ user_id: 1 })
drop table users	db.users.drop()
insert into users(user_id,age, status)values ("bcd001",45,"a")	db.users.insert({ user_id: "bcd001", age: 45, status: "a" })
select * from users where status = "a"	db.users.find({ status: "a" })

II. PROPOSED METHODOLOGY

The main aim of this study is the realization of a methodology for facilitating the migration from a RDBMS (MySQL) to NoSQL (MongoDB). The main steps of the methodology are visually displayed in Fig. 2, which, as highlighted in black circles, explain how the methodology works. To present each step, we will use an example of a database representing the orders of a company in Fig. 3.

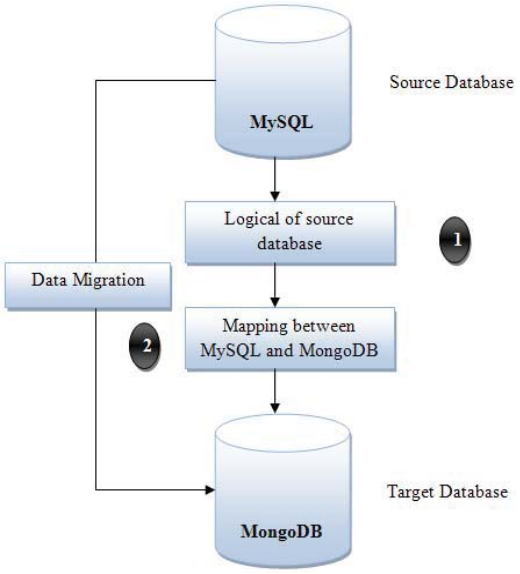

Fig. 2 A methodology for data migration

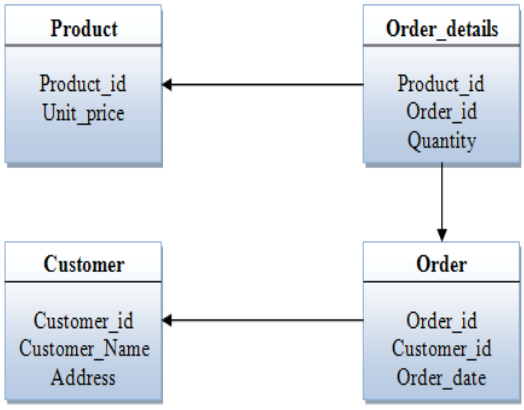

Fig. 3 Relational database logical structure

A. Loading the Logical Structure of the Source Database

In the beginning it is necessary to specify the relational database source. Then, we connect to source database to obtain all information about type and version. We also need to specify all the information about NoSQL target database. Furthermore, in this step we obtain a representation of the relational model of the source database. For this we need to get the names of tables, their attributes and relationships. The information on the relationships can be retrieved from the primary constraint and foreign keys for each table. Fig. 3 is an instance of our model representing a portion of our source database.

B. Mapping Between Relational Model and MongoDB Model

This step is dedicated to define the mapping between the relational model of MySQL and document-oriented model of MongoDB. It consists in defining which attributes of the relational model of the source database will be linked to the attributes of the target model. Loading database tables and their attributes can be implemented in different ways. One of them is to use JDBC driver. During the migration of relational model, the system will propose a set of suitable tables to migrate to the target database. The proposal of suitable tables will be based on the type of the MySQL of the source database.

The main concern of those coming from a relational background is the absence of JOINs in NoSQL databases. As demonstrated below, the document model makes JOINs redundant in many cases. In Fig. 4, the relational database uses the "Customer_ID" field to join the "Customer" table with the "Order" table to enable the application to report each order to the right customer. Using the document model, embedded sub-documents and tables effectively pre-join data by aggregating related data within a single data structure. Columns and rows that were traditionally normalized and distributed across separate tables can presently be stored together in a single document, eliminating the need to JOIN separate tables when the application has to retrieve complete records [10].

Modeling the same data in MongoDB enables us to create a schema in which we embed an array of sub-documents for each order directly within the customer document (Fig. 5). In this example, the application relies on the RDBMS to join four separate tables. With MongoDB, all of the data is aggregated within a single document, linked with a single reference to a customer document (Fig. 6) [10].

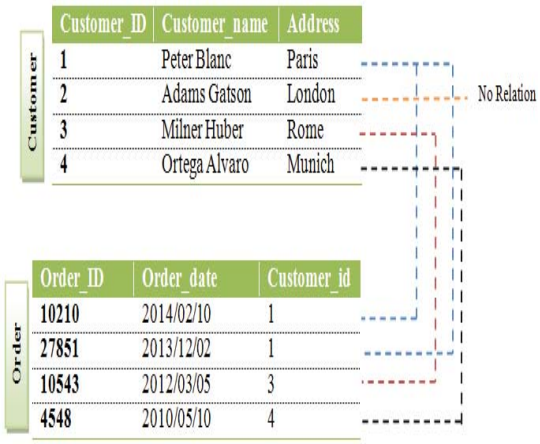

Fig. 4 Relational Schema, Flat 2-D Tables (Customer and Order)

```
{
Order_ID: "X192",
Order_Date: 20150210,
customer:[
        { Customer_ID: "54",
        Customer_Name: "Paul",},
        { Adress: "London",
        Postal: 23090,}
]
}
```

Fig. 5 Richly Structured MongoDB Document

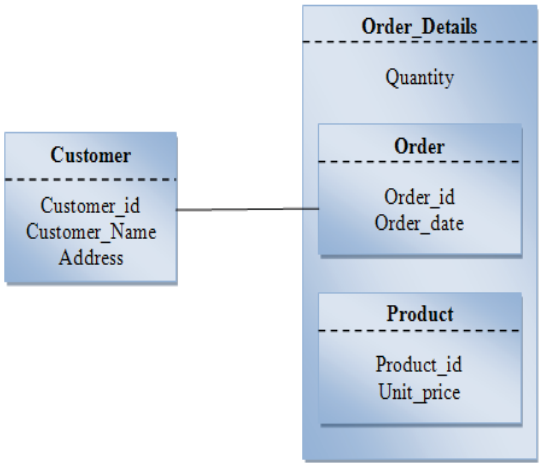

Fig. 6 The MongoDB model result of the mapping from relational model

III. Proposed Software

In order to test the validity of the proposed methodology, we develop a software prototype which is programmed by

using JAVA on a PC platform. The operation sequence will be demonstrated in the following paragraphs, through the use of several screenshots.

Initially, the user must connect to MySQL system through the interface shown in Fig. 7 for choosing the source database which will migrate to MongoDB.

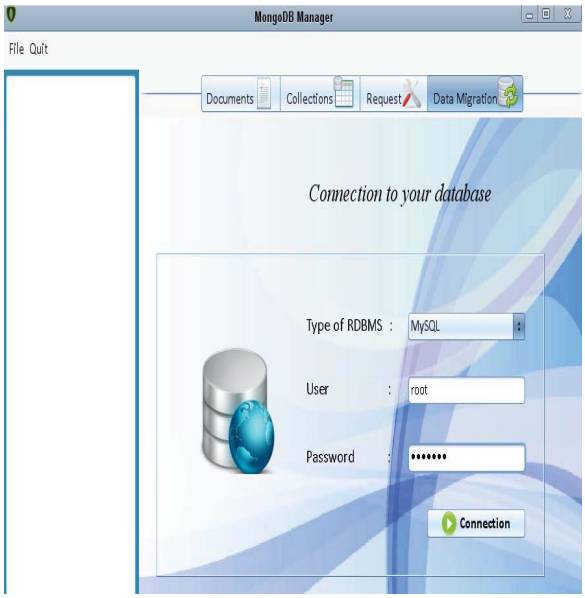

Fig. 7 Connection to source database

Next, Fig. 8 provides a set of tables of source database to the user, which he will choose in order to consult or migrate. If the migrated table is join to another table, both will be migrated automatically.

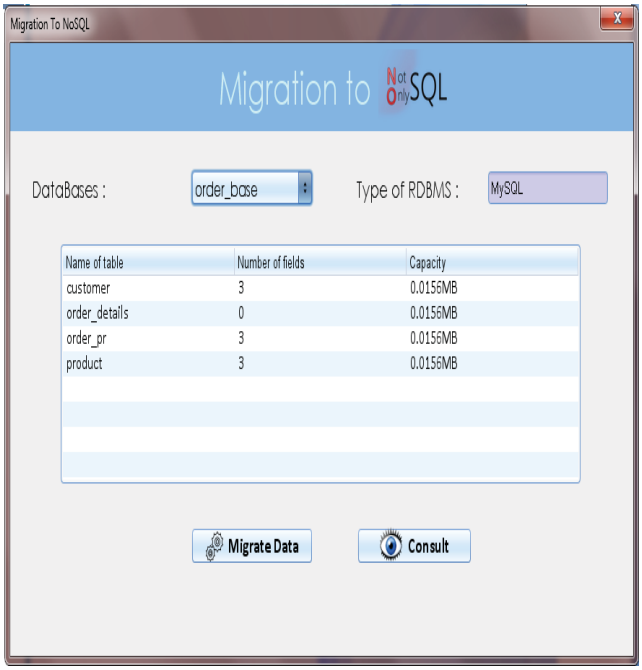

Fig. 8 Migration to MongoDB

Finally, the data are inserted in MongoDB. Fig. 9 presents data migrated in collection forms where the user can access to all the features of MongoDB using the tabs presented in this interface: Documents, Collections, and Requests.

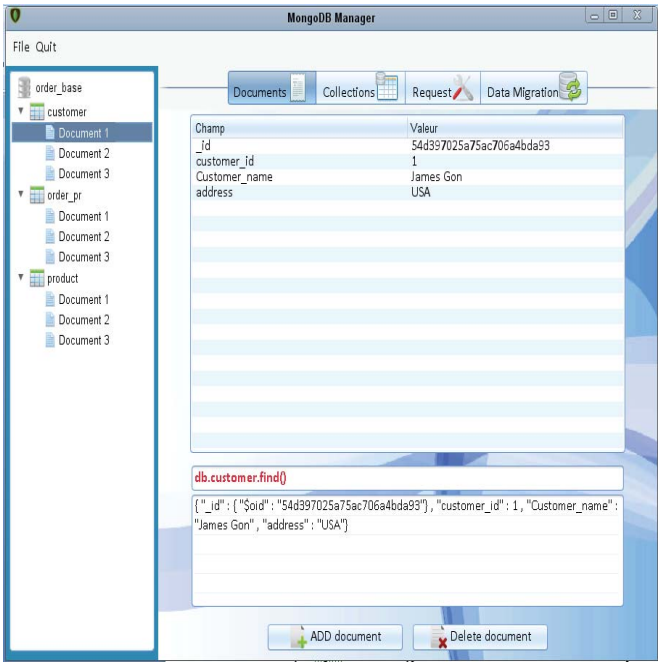

Fig. 9 Result of migration

IV. Conclusion

In this paper, we have focused on data migration from relational to NoSQL database. We have also discussed that NoSQL is better and faster than RDBMS. Next, we presented methodology for data migration from MySQL as RDBMS to MongoDB as NoSQL oriented-document database. During data migration, the model structure of the target (NoSQL) was automatically created from relational schema. For illustration, we propose an application developed on JAVA that is based on the proposed methodology.

As a future direction, the software that we developed could be enhanced by other features and improvements, such as: To cover all types of databases available as RDBMS or NoSQL for migrating from both of them, not just the MySQL and MongoDB.

Acknowledgment

The authors wish to acknowledge the contributions of other members of the department of computer science for their helpful discussions and the availability of all resources that have helped make this work in the best conditions. They also wish to thank Mr. Redouane Boulguid for pointing out many English corrections that lead to the improvement of the paper. The authors would also like to thank the reviewers for their remarks and suggestions.

REFERENCES

[1] B. Walek, K. Cyril, "A methodology for Data Migration between Different Database Management Systems". World Academy of Science, Engineering and Technology Vol:6 2012-05-23
[2] B. Walek, K. Cyril, "Data Migration between Document-Oriented and Relational Databases". World Academy of Science, Engineering and Technology Vol:6 2012-09-22
[3] M. Alam, and Krishan Wasan S. "Migration from Relational Database into Object Oriented Database". Journal of Computer Science 2 (10): 781-784, 2006.
[4] X. Lixian, L. Yanhong, "Design and application of data migration system in heterogeneous Database". International Forum on Information Technology and Applications 2010.
[5] N. Cory, L. Travis, I. Reenu, H. Gary, "Nosql Vs Rdbms - Why There Is Room for Both". Proceedings of the Southern Association for Information Systems Conference, Savannah, GA, USA March 8th–9th, 2013.
[6] M.A Mohamed, O.G Altrafi, M.O Ismail, "Relational vs. NoSQL Databases: A Survey". International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 03 – Issue 03, May 2014.
[7] R. Cattell, "Scalable SQL and NoSQL Data Stores". SIGMOD Record, December 2010 (Vol. 39, No. 4)
[8] MongoDB, (2014) The MongoDB 2.6 Manual: http://docs.mongodb.org/manual
[9] NoSQL, (2014) Making sense of NoSQL, A guide for managers and the rest of us: Dan McCreary et Ann Kelly. 2014.
[10] Migration Guide, (2015) RDBMS to MongoDB Migration Guide Considerations and Best Practices a MongoDB White Paper February 2015.