ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse234922020.pdf
https://doi.org/10.30534/ijatcse/2020/234922020

MQL2SQL: A Proposal Data Transformation Algorithm from MongoDB to RDBMS

Jabrane Kachaoui[1], **Abdessamad Belangour**[2]
[1]Hassan II University, Morocco, jabrane2005@gmail.com
[2]Hassan II University, Morocco, belangour@gmail.com

ABSTRACT

As Big Data applications grow, many existing systems expect expanding their service to cover data dramatic increase. New software development systems are no longer working on a single database but on current multidatabases. These distributed data sources are under the name of NoSQL (Not only Structured Query Language) databases. Several companies try taking advantages from these technologies but without leaving their traditional systems. Especially, Data Warehouses (DW) are conceived based on users' feedbacks. To allow and support this integration, a mechanism that takes data from NoSQL databases and stores it into relational databases is needed to have great added value without impacting organization existing systems. This paper proposes an integration algorithm to support hybrid database architecture, including MongoDB and MySQL, by allowing users to query data from NoSQL systems into relational SQL (Structured Query Language) systems.

Key words: Big Data, NoSQL, MongoDB, Data Warehouse.

1. INTRODUCTION

Today, DW is a necessity for any business. It allows fast and reliable access to various important data and helps in making decisions within the company. It permits a proper functioning through enhanced decision analysis [1]. To respond to market requirements and technological developments that are continuously increasing, DW must implies an integration in this era of massive data storage otherwise called Big Data. Several aspects ranging from server upgrades to adding new platforms for the extended DW perform this combination [23]. This could initiate the use of certain functions that were previously untapped, such as the In-Memory database, real-time functions and virtualization [2].
It would be absolutely absurd to keep this data storage system identical while the evolution around it is constant. This will reduce the efficiency and reliability of the system, which will adversely affect decision analysis. There would thus be several stages for DW integration according several studies about business concern, technical performance scale, improved analytics, new data-driven practices and real-time operations [21].

The quality of data integration within DW is now recognized as the main factor conditioning the success of decision-making system. However, this essential point should not mobilize all companies' efforts. The whole project is complex. It would therefore be unwise not to devote as much attention to the tricky issue of organizing decision-making bases. The existing system is well established for ease of use and access to get adequate information. Indeed, users never have the time and the energy necessary to dig into databases of hypothetical information. Access to information in a reasonable time actually reflects the degree of convenience (and use) of decision-making system. To put it simply, the easier the information will be to access, the more the instrument will be used and therefore, the more it will be called upon to develop. This is no more and no less the factor defining the overall performance of decision support system [3].
Several researches have been conducted for taking advantage of DW from large amount of data stored in Data Lake (DL) [4]. A first study was made by designing a complete architecture that routes data from DL to DW [5]. This architecture focused on a data pre-processing aspect and ontologies, for the aim of data categorization in many clusters using K-means algorithm [6], [7], [22]. Data is stored in structured format using MongoDB characteristics.
This paper aims to propose an automatic algorithm mapping of MongoDB NoSQL databases to relational databases. It is outlined as follows. Next section discusses some related works by describing their proposal approaches in this context. The third section, defines major concepts contributing in research's achievement. A proposal algorithm, Mongo Query Language To Structured Query Language (MQL2SQL) for automatic mapping of MongoDB NoSQL databases to relational databases is proposed in fourth section. Finally, the last section draws conclusion and future works.

2. RELATED WORKS

Several researches were carried out to perform data transformation. They focused on three categories: NoSQL databases modelling, conversion schema and data transfer.
Gansen Zhao et al. presented a MongoDB relational model and achieved the query operation under relational algebra compromising single and several sharing situations even

though MongoDB is a document-oriented database with no predefined schema. Furthermore, MongoDB can handle relational computation as relational databases. Therefore, they concluded that data can be transferred easily and securely from relational database to MongoDB [8]. In MongoDB, it is able to model relationships between documents through referenced and embedded document.

Anuradha Kanade et al, have discussed the performance variation as well as the technique change and they have led several experiences looking for the importance of normalization and integration in reducing execution time of queries in MongoDB [9]. With NoSQL databases evolution, various tools have been emerged for data transformation from relational databases to NoSQL databases like Apache Sqoop [10] and Pentaho Data Integration (Kettle) [11].

Apache Sqoop is a software conceived for data transfer between Apache Hadoop and structured data. It is employed for data import and export from relational database such as MySQL, Oracle to Hadoop HDFS and vice versa.

Wu-Chun Chung et al, have developed a framework entitled JackHare including SQL query compiler, JDBC driver and a Systematic method based on HBase and Hadoop for storing data contained in relational database by MapReduce for unstructured data processing. This framework aimed handling large-scale data [12].

Tianyu Jia et al, have designed a transformation model approach and data migration from RDBMS (Relational Database Management System) to MongoDB. To prevent data redundancy and optimize performance, the transformation algorithm model optimizes only specific database tables using action tags and descriptions to describe relational schema limitation [13].

In summary, many approaches and algorithms focused on schema conversion, while others on data transformation from RDBMS to NoSQL databases and performance improvement to complete high scalability and availability. However, to authors’ knowledge, no study was conducted transforming data from NoSQL databases to relation databases.

3. PRELIMINARIES

3.1 NoSQL databases

NoSQL databases mark a rather brutal break with the way of designing data schemas that have been designed for a few decades. Specifically dedicated to Internet-oriented applications, NoSQL databases overcome the difficulties of managing relational databases that are a little too large and spread over several machines.

The NoSQL database Type, a fortiori the "document" oriented bases, abandon the strong point of relational bases, which is the notion of relationships between elements, to focus on the notion of "document". NoSQL databases are much more flexible and much more scalable. The organizational structure is no longer linked to a relational pattern that is difficult to modify, database can therefore grow without constraint.

On the other hand, "document" orientation facilitates database deployment on multiple machines. In automatic of course. The developer is not concerned with the location of documents, split or not [14].

There are also NoSQL databases of type "columns" and databases of type "graph". Column-type databases are an excellent solution for carrying out massive analyses. However, they are more complex to use. The graph-type databases, which are more difficult to grasp, are, as their denomination indicates, more suited to solving questions of network organization (structure in arcs and nodes), which are particularly useful for managing the notions of belonging to social groups for example. It suffices to follow the graph by browsing the nodes to perceive all of the links and more easily access a specific element [15].

3.2 MongoDB

MongoDB is a DBMS(Database Management System), like MySQL or PostgreSQL, but whose mechanism is completely different. It does not need to create a relational table schema and create complex SQL queries. Thanks to MongoDB, it will be able to store data much like it would in a JSON (JavaScript Object Notation) file. That is to say, a sort of giant dictionary made up of keys and values. This data can then be exploited by JavaScript, directly integrated into MongoDB, but can also be exploited by other languages such as Python [16].

Document manipulation is a main objective of MongoDB, as it provides various frameworks such MapReduce and ways of documents interaction. Documents can be iterated, queried and sorted with cursors, aggregated by other operations. The document changes are maintained to be atomic. MongoDB supports several programming languages.

3.3 RDBMS

A relational database is a type of database where data is linked to other information within databases. Relational databases are made up of a set of tables that can be accessed and rebuilt in different ways, without the need to rearrange these tables in any way. SQL is the standard interface for a relational database. Its statements are used both to interactively query data in relational database and to collect data for reporting.

RDBMS makes it possible to highlight relationships between data. This data is organized in a table in rows and columns in order to be accessible. Tables contain all information about relationships between different data. In the Product table example, each row is a specific product and columns list product attributes, such as colour, size, etc. [17].

4. DATA TRANSFORMATION ALGORITHM

4.1 MQL2SQL proposal data model

MongoDB is a high performance scalable open source NoSQL database. MongoDB uses document store rather than two-dimensional table structures. 10gen Company conceived it [16]. The differences between MongoDB and normal relational databases is represented in the following table.

Table 1: Differences between RDBMS and MongoDB

RDBMS	**MongoDB**
Database	Database
Table	Collection
Tuple/Row	Document
Column	Field
Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

- Collections in MongoDB is equivalent to tables in RDBMS.
- Documents in MongoDB is equivalent to rows in RDBMS.
- Fields in MongoDB is equivalent to columns in RDBMS.

Data management in MongoDB is flexible. This flexibility gives the choice of data modelling according to its performance requirements. MongoDB collections do not require a document structure. Each document can have different fields from other documents. One document with four fields, another with seven for example.

There are two ways in MongoDB allowing linking documents, references and embedded documents. The first method, documents are linked by links, that means that each document contains a reference to another document. This is the normalized data model. Otherwise, in embedded documents, the linked data is stored in a single document. MongoDB treats them as subdocuments.

This denormalization of the given models allows manipulating linked data with a single operation, that is to say a single request.

The purpose of the proposal model in to propose a unified data model that combines both of MongoDB data model to a relational data model.

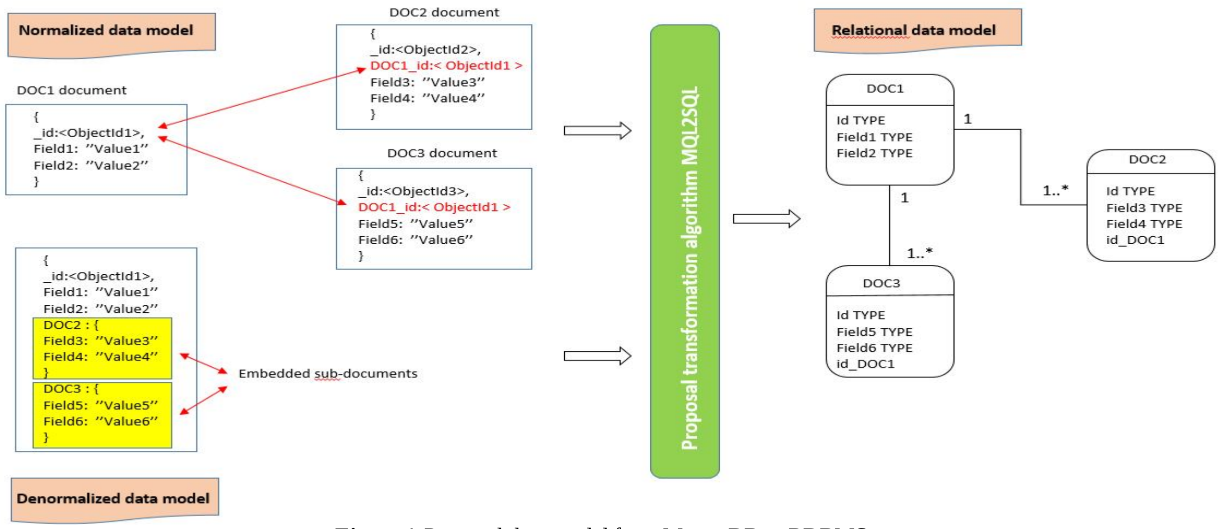
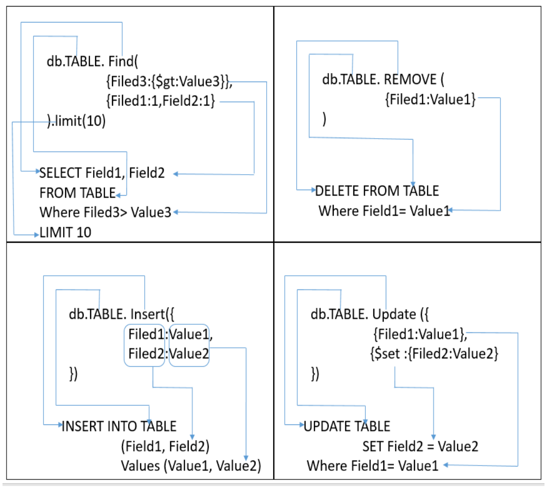


Figure 1: Proposal data model from MongoDB to RDBMS

4.2 Data transformation

In order to provide MongoDB functionality for SQL, it is fundamental to implement the transformation algorithm MQL2SQL. For MongoDB and SQL, there are dissimilarities in concepts and terms between them, and they are not the same in data aggregation operations. The comparison between SQL and the
MQL is illustrated in Figure 2, these operations are equivalent to CRUD (Create, Read, Update and Delete).



```
db.TABLE. Find(
        {Filed3:{$gt:Value3}},
        {Filed1:1,Field2:1}
).limit(10)

SELECT Field1, Field2
FROM TABLE
Where Filed3> Value3
LIMIT 10
```

```
db.TABLE. REMOVE (
        {Filed1:Value1}
)

DELETE FROM TABLE
Where Field1= Value1
```

```
db.TABLE. Insert({
        Filed1:Value1,
        Filed2:Value2
})

INSERT INTO TABLE
        (Field1, Field2)
        Values (Value1, Value2)
```

```
db.TABLE. Update ({
        {Filed1:Value1},
        {$set :{Filed2:Value2}
})

UPDATE TABLE
        SET Field2 = Value2
Where Field1= Value1
```

Figure 2: Comparison between SQL and MQL

As illustrated in Figure 2, the difference appears in grammatical format. For example, the SQL syntax for inserting a row record is INSERT INTO {Table_Name} {Field_Name_List} VALUES (Field_Value_List), while in MQL syntax for the same operation is "db". + {Table_name} + ".insert ({" + ({Field_name}: {Field_value}) [, ...] + "})". Here, this query operation is taken as an example to describe the MQL2SQL transforming method.

4.3 Proposal approach

In RDBMS, MongoDB remains a database. MongoDB collection is mapping to a relational table and documents to tuples. Before starting defining algorithm of mapping, Different relationship between entities must be firstly defined. This paper focuses on the three associations between entities 1:1, 1:M and N:M.
The 1:1 describes relationship between two entities. For example, a Student has a single Course relationship. A student studies a single course and a course only studied by single student.

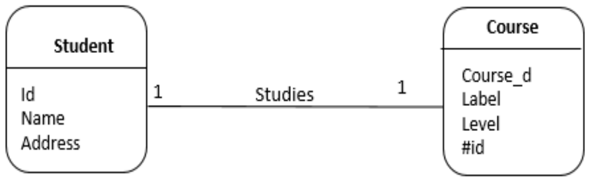


Figure 3: One to one RDBMS association

MongoDB can model the 1:1 association in two ways. The first model is an embedded relationship as single document; the second is a linked document in separate collection [18]. The code bellow describes both methods of the one to one association modelling:
Student document:

```
{Name:"Jabrane Kachaoui",
 Address:"Casablanca"}
```

Course document:

```
{Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced"}
```

The first model is to embed the Course document in Student document. An example of Course document with embedded course:

```
{Name:"Jabrane Kachaoui",
 Address:" Casablanca",
Course:{
Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced"}
}
```

The second model is to link Course and Student document.

An example of Student document:

```
{id:1,
Name:" Jabrane Kachaoui",
Address:" Casablanca"}
```

An example of Course document with Foreign Key:

```
{Student_id:1,
   Label:"Mathematics",
   Nbr_hours:100,
   Level:"Advanced"}
```

From the two models, it is noticed that the second model of linked documents is similar to how traditional relational databases store data. This study is focused on this model to project data from MongoDB to RDBMS in one to one association [19].
Now, supposing that a student can study one or more courses. For the moment, the case where a course is studied by lot of students is excluded. Then, the following two sentences are constructed:

- One course is studied by a single student.
- A student studies several courses.

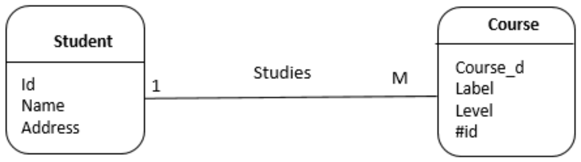


Figure 4: One to many RDBMS association

The 1:M association can be modelled in several methods using MongoDB. The first model is to embed course in student:

```
{Id:1,
Name:"Jabrane Kachaoui",
Address:" Casablanca",
Course:[
{Course_id=20,
Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced"},
{Course_id=20,
Label:"Physics",
Nbr_hours:150,
Level:"Meduim"}]}
```

The second model is to link courses to student using a foreign key. An example of Course documents with Foreign Keys:

```
{id:1,
Course_id=20
 Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced"}
{id:1,
 Course _id:21,
Label:"Physics",
Nbr_hours:150,
Level:"Meduim"}
```

The third model is bucketing. It is a combination of the two models described above. Mainly, it tries to balance the embedding model rigidity with linking model flexibility. For this example, course is divided into buckets with a maximum of 7 courses in each bucket:

{id:1,
Page:1,
Count:7,
Courses:
[{Course_id:2O,
Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced"},
{ Course_id:21,
Label:"Physics",
Nbr_hours:150,
Level:"Meduim"},…]}

The main advantage of using buckets, is to perform a single read to return 7 courses at a time, allowing for efficient pagination. When there is possibility of dividing documents into discreet batches, it is better to consider bucketing to accelerate document recovery.

A N:M association seems to be 1:M in both directions, that means it is a many to many relationship. To manage such relationship, an additional table called "Studies" is created, which contains the primary keys of both Student-Course couples, using the corresponding codes. This procedure is illustrated in the example below.

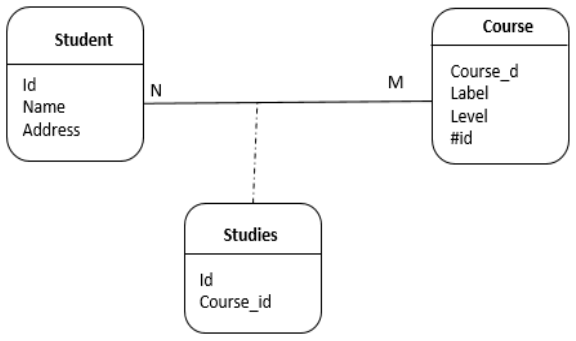


Figure 5: Many to many RDBMS association

In MongoDB, this situation is represented in many ways. The first is titled Two Way Embedding. In this method, the Course foreign keys are included under the Course field in the Student document. Mirroring Student document, for each Course, the Student foreign keys are included under Student field in Course document.

An example of Student documents:

{id:1,
Name:"Jabrane Kachaoui",
Address:"Casablanca",
Course:[1,2]}
{id:2,
Name:"Jean Paul",
Address:"Paris",
Course:[2]}

An example of Course documents:

{Course_id:20,
Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced",
Student:[1]}
{ Course _id:21,
Label:"Physics",
Nbr_hours:150,
Level:"Meduim",
Student:[1,2]}

The second way of modeling N:M association is the One Way Embedding. This method optimizes the read performance of the N:M by setting references in one side of the association. It this case, A Course can belong to few Categories and a Category can have many Courses.

An example of Category documents:

{id_cat=1,
name="Theorical"}
{id_cat=2,
name="Practical"}

An example of a Course document with foreign keys for Categories:

{Course_id:20
Label:"Mathematics",
Nbr_hours:100,
Level:"Advanced",
Categories:[1],
Student:[1]}
{Course_id:21
Label:"Physics",
Nbr_hours:150,
Level:"Meduim",
Categories:[1,2],
Student:[1,2]}

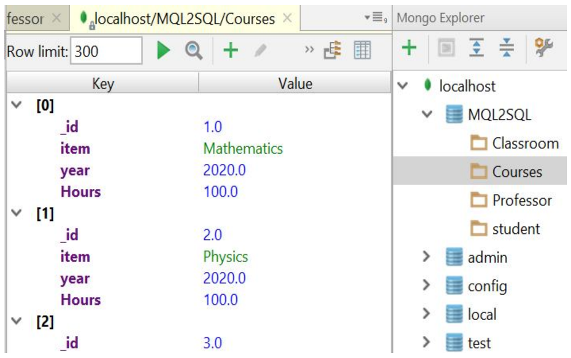
The purpose for selecting to embed all references to categories in the courses is due to the fact that lot more courses in a category than categories in a course. To choose one of the two models, user must make the maximum size of N and M. For example if N is a maximum of 3 categories for a course and M is a maximum of 10000 courses in a category, One Way Embedding is more adapted to this context. If N is a maximum of 3 and M is a maximum of 5 then Two Way Embedding can work well.

4.4 Algorithm Implementation

MQL2SQL is a framework implementing an algorithm of mapping of MongoDB to relational databases. This algorithm was developed and tested using MySQL databases as a RDBMS and MongoDB as a NoSQL database.

For examples, a database MQL2SQL is used that contains four Collections: Student, Professor, Classroom and Course. The relationships are:

- 1:M between Professor and Course;
- 1:M between Course and student;
- 1:M between Student and Course;
- 1:1 between Professor and Classroom;



Key	Value
[0]	
_id	1.0
item	Mathematics
year	2020.0
Hours	100.0
[1]	
_id	2.0
item	Physics
year	2020.0
Hours	100.0
[2]	
_id	3.0

Figure 6: MongoDB database

MQL2SQL implementation steps are presented next:

1. Creating the MySQL database. The user must specify the MongoDB database that will be represented in MySQL. The database is created with the following SQL command:

>CREATE DATABASE DATABASE_NAME;

2. Creating tables in the new MySQL database. The algorithm verifies for each collection in what relationships is involved, if it is referred by other collection.

i. If the collection is not referred by other collections, it will be represented by a new table.

ii. If a collection is referred by other collection, table is created without foreign keys, but is referred by another table.

iii. If a collection refers to another collection. The framework uses the linking method and translates it to foreign key concept. A table is created with one foreign key and is referred by other table.

iv. If a collection is embedded into another collection from the one side of the relationship. The framework uses one way embedding model and translates it to foreign key concept. A table is created with one foreign key but not referred by any table.

v. If each collection refers to other collection, framework uses the two way embedding model. A table is created with two foreign keys and is not referred by any table.

vi. If a collection refers to three or more collections, algorithm uses the linking model. A table with these three or more foreign keys is created. It is the result of a N:M ternary relationships.

For extracting the name of MongoDB collection, the next command is used:

>use MQL2SQL //to switch to the desired database
>show collections //to retrieve all database collections

For metadata, MongoDB uses GridFS that relies on two collections to manage large files: the first, called chunks, hosts documents that make up the file while the second, called files, and contains metadata associated with the file. In chunks, there is particularly the sequence number of the piece of file and its content in BSON (Binary JSON) form and in files, information such as the date of download, the size of each piece of file or the name of the file split into several chunks [20].

With these data, framework can establish steps already described (2.i-2.vi). Then, collections become relational tables in MySQL. For the MongoDB database from figure the mapping according to algorithm is presented next.

Professor collection is linked with Classroom Collection. Professor and Classroom become tables with one to one association (step 2.ii). Student collection refers Courses collection and Courses collection refers Student collection. Student and Courses collection becomes tables (step 2.iii). A joined table will be created using two way embedding model (step 2.v). Five relational tables will represent the four MongoDB collections.

5. CONCLUSION AND FURURE WORKS

This paper proposes an effective transformation approach from relational databases to NoSQL data-stores including MySQL and MongoDB in this case of study, which allows users to query data from NoSQL systems to relational SQL systems. The study describes in detail the transformation method between SQL and MongoDB, structures of the two kind's modification. Finally, transformation was evaluated on small size database using the proposal algorithm. This is the result of an advanced study on the related topic, which fills the gap overlooked by relevant scholars in this field to make a little contribution of data transformation.

Future works involve the optimization of algorithm and supporting for NoSQL's transaction feature. In addition, authors intend to integrate more NoSQL databases transformation.

REFERENCES

1. H. Khazaei et al. **How do I choose the right NoSQL solution? A comprehensive theoretical and experimental survey**. AIMS' Journals,2015.
2. J. Kachaoui and A. Belangour. **An Adaptive Control Approach for Performance of Big Data Storage Systems.***In Proc. International Conference on Advanced Intelligent Systems for Sustainable Development, 2019, pp 89-97.* https://doi.org/10.1007/978-3-030-36674-2_9
3. P. Ripon, and A.Arif. **Big Data: The V's of the Game Changer Paradigm**, *in Proc. IEEE 18th International Conference on High Performance Computing and Communications, 2016.*
4. J. Kachaoui and A. Belangour. **A Multi-criteria Group Decision Making Method for Big Data Storage Selection**,*in Proc. International Conference on Networked Systems, 2019, pp 381-386.* https://doi.org/10.1007/978-3-030-31277-0_25
5. J. Kachaoui and A. Belangour. **Challenges and Benefits of Deploying Big Data Storage Solution**, *inProc. of the New Challenges in Data Sciences: Acts of the Second Conference of the Moroccan Classification Societ, Article No.: 22,2019, pp 1–5.* https://doi.org/10.1145/3314074.3314097
6. J. Kachaoui and A. Belangour. **Enhanced Data Lake Clustering Design based on K-means Algorithm**.

International Journal of Advanced Computer Science and Applications, in progress.

7. J. Kachaoui, J. Larioui and A. Belangour. **Towards an Ontology Proposal Model in Data Lake for Real-time COVID-19 Prevention Cases**. International Journal of Emerging Technologies in Learning (iJET), unpublished.
8. Z. Gansen, L.Qiaoying, L.Libo, L.Zijing. **Schema Conversion Model of SQL Database to NoSQL**. Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2014.
9. A. Kanade, A. Gopal, and S. Kanade. **A study of normalization and embedding in mongodb**. in Advance Computing Conference (IACC), IEEE International. IEEE, 2014,pp. 416–421.
10. Apache Software Foundation http://sqoop.apache.org/
11. M. Casters, R. Bouman, and J. Van Dongen. **Pentaho Kettle solutions: building open source ETL solutions with Pentaho Data Integration**. John Wiley & Sons, 2010.
12. Chung, Wu-Chun, LIN, Hung-Pin, Chen, Shih-Chang, et al. **JackHare: a framework for SQL to NoSQL translation using MapReduce.** Automated Software Engineering. vol. 21, no 4, 2014, p. 489-508. https://doi.org/10.1007/s10515-013-0135-x
13. T. Jia, X. Zhao, Z. Wang, D. Gong, G. Ding. **Model transformation and data migration from relational database to mongodb**. IEEE International Congress on Big Data (BigData Congress), 2016, pp. 60-67.
14. S. H. Aboutorabia, M. Rezapourb, M. Moradic, N. Ghadirid. **Performance evaluation of SQL and MongoDB databases for big e-commerce data**. International Symposium on Computer Science and Software Engineering (CSSE), 2015.
15. L. Stanescu, M. Brezovan, D. D. Burdescu. **An algorithm for mapping the relational databases to mongodb - a case study**. International Journal of Computer Science and Applications, 2017.
16. S. Pore, Swalaya B. Pawar. **Comparative Study of SQL & NoSQL Databases**. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 5, 2015.
17. J. R. Lourenço et al. **Choosing the right NoSQL database for the job: a quality attribute evaluation**. Journal of Big Data,2015,pp 2-18. https://doi.org/10.1186/s40537-015-0025-0
18. J. Celko. **What Every SQL Professional Needs to Know about Non-Relational Databases**. 1st Edition. Elsevier, USA, 2014.
19. A. Krisciunas, **Benefits of NoSQL**: https://www.devbridge.com/articles/benefits-of-nosql/, 2014.
20. F. Toufik, M. Bahaj. **Model Transformation from Object Relational Database to NoSQL Document Database**.*In Proc. of the second International Conference on Networking, Information Systems & Security, Article No.: 49, 2019, pp 1–5.*
21. M.Burawis and R. J. Cabauatan. **Query Optimization: Fund Data Generation Applying NonClustered Indexing and MapReduced Data Cube Numerosity Reduction Method.** International Journal of Advanced Trends in Computer Science and Engineering 9(1.1 S I),2020, pp102-109. https://doi.org/10.30534/ijatcse/2020/1991.12020
22. J. E. Sabugaa and G. S. Lahayon. **Digital Data Classification and Extraction for Records Management of PAPS and PACS Documents**. International Journal of Advanced Trends in Computer Science and Engineering Volume 9, No.1.1, 2020, pp 272-277. https://doi.org/10.30534/ijatcse/2020/4891.12020
23. J. Kachaoui and A. Belangour. **From Single Architectural Design to a Reference Conceptual Meta-Model: An Intelligent Data Lake for New Data Insights**. International Journal of Emerging Trends in Engineering Research, in progress.