Journal of Ubiquitous Systems & Pervasive Networks
Volume 8, No. 2 (2017) pp. 15-20

From Relational Database to Big Data: Converting Relational to graph database, MOOC database as example

Yemna Sayeb*, Radhouane Ayari, Sarra Naceur, Henda Ben Ghezala

University of Manouba, RIADI, National School of computer science, Manouba campus, Manouba 2010, Tunisia

Abstract

Existing graph database management systems provide an efficient solution to data storage where graph models are widely used, the conversion of an existing application from a relational to graph data can be convenient but it is usually a hard task for database administrators. In this work, we propose a conversion tool from a relational to graph database. The approach supports the conversion of the schema and the data.

Keywords: *NoSQL; Migration; Relational Database; Graph database.*

1. Introduction

Face to the growing need for performance and availability relational databases quickly reach their limits and adding hardware servers does not increase performance enough. Following this, new technologies have emerged such as NoSQL databases which are radically changing the architecture of the database. With these technologies we can increase performance and improve availability of services [1]. So, several projects have found themselves facing the necessity to migrate to this new technology.
Thus, Google has migrated to NoSQL in 2004 with the engine BigTable. It was followed by the giants of the social web to know Facebook, Twitter and LinkedIn. These companies have migrated all or part of their information system of relational databases to NoSQL databases. Also, this technology is a new concept and there are many research work improving existing approach and migration tools. In this case we have seen useful to propose an approach to migrate a relational database to a NoSQL database. To do this we began by studying the limitations of relational databases which are facing problems of large volumes of data, then we present advantages of NoSQL databases, finally we present an approach to migrate a relational to a NoSQL database.

2. Big Data advantages [2]

With Big Data databases, enterprises can save money, grow revenue, and achieve many other business objectives, in any vertical. It might allow a company to collect billions of real-time data points on its products, resources, or customers and then repackage that data instantaneously to optimize customer experience or resource utilization.
Big data technologies can replace highly-customized, expensive legacy systems with a standard solution that runs on commodity hardware. And because many big data technologies are open source, they can be implemented far more cheaply than proprietary technologies.
Big data can help businesses act more nimbly, allowing them to adapt to changes faster than their competitors. And it allows businesses and other organizations to more rapidly and accurately respond to customer demand by increasing the amount of data shared within the organization and the speed with which it is updated.

3. NOSQL Databases

3.1. What is a NoSQL (Not Only SQL) Database? [3]

A NoSQL (Not-only-SQL) database is one that has been designed to store, distribute and access data using methods that differ from relational databases (RDBMS's). NoSQL technology was originally created and used by Internet leaders such as Facebook, Google, Amazon, and others who required database management systems that could write and read data anywhere in the world, while scaling and delivering performance across massive data sets and millions of users.
Today, almost every company and organization has to deliver cloud applications that personalize their customer's experience with their business, with NoSQL being the database technology of choice for powering such systems.
A NoSQL database offers the following benefits over other database management system:

[*] Yemna SAYEB. Tel.: +216 98 521 335
E-mail: ysayeb.recherche@gmail.com

DOI: 10.5383/JUSPN.08.02.003

- Continuously Available: A database that stays online even in the face of the most devastating infrastructure outages.
- Geographically Distributed: Fully active data, everywhere you need it.
- Operationally Low Latency: Response times fast enough for most intense operational cloud applications.
- Linearly Scalable: Predictably scale to meet the current and future data needs of cloud applications.
- Immediately Decisive: Full range of data manipulation capabilities tightly integrated into a single system.

As enterprises shift to the Digital Economy enabled by cloud, mobile, social media, and big data technologies, developers and operations teams have to build and maintain web, mobile, and Internet of Things (IoT) applications faster and faster, and at greater scale. NoSQL is increasingly the preferred database technology to power today's web, mobile, and IoT applications.
With NoSQL, enterprises are better able to both develop with agility and operate at any scale, and to deliver the performance and availability required to meet the demands of their business objectives.
To see closely the NoSQL databases we have seen useful to study the types of NoSQL database.

3.2. NoSQL Databases types [4]

NoSQL databases can broadly be categorized in four types:

3.2.1. Key-value databases

3.2.1.1 What is key-value database?

The key-value data stores are simplistic, but are quiet efficient and powerful model. It allows the user to store data in a schema less manner. The data is usually some kind of data type of a programming language or an object. The data consists of two parts, a string which represents the key and the actual data which is to be referred as value thus creating a "key-value" pair. These stores are similar to hash tables where the keys are used as indexes, thus making it faster than RDBMS Thus the data model is simple: a map or a dictionary that allows the user to request the values according to the key specified. The modern key value data stores prefer high scalability over consistency. Hence ad-hoc querying and analytics features like joins and aggregate operations have been omitted. High concurrency, fast lookups and options for mass storage are provided by key-value stores. One of the weaknesses of key value data sore is the lack of schema which makes it much more difficult to create custom views of the data. Key value data stores can be used in situations where you want to store a user's session or a user's shopping cart or to get details like favorite products. Key value data stores can be used in forums, websites for online shopping etc. Although key-value data stores existed for long time ago, the development of large number of recent key value data store was influenced by the introduction of Amazon's Dynamo.

3.2.1.2 Scalability and reliability

Key-value stores scale out by implementing partitioning (storing data on more than one node), replication and auto recovery. They can scale up by maintaining the database in RAM and minimize the effects of ACID guarantees (a guarantee that committed transactions persist somewhere) by avoiding locks, latches and low-overhead server calls.

3.2.2. Document Databases

3.2.2.1 What is Document database?

Document Store Databases refers to databases that store their data in the form of documents. Document stores offer great performance and horizontal scalability options. Documents inside a document-oriented database are somewhat similar to records in relational databases, but they are much more flexible since they are schema less. The documents are of standard formats such as XML, PDF, JSON etc. In relational databases, a record inside the same database will have same data fields and the unused data fields are kept empty, but in case of document stores, each document may have similar as well as dissimilar data. Documents in the database are addressed using a unique key that represents that document. These keys may be a simple string or a string that refers to URI or path. Document stores are slightly more complex as compared to key-value stores as they allow to encase the key-value pairs in document also known as key-document pairs. Document oriented databases should be used for applications in which data need not be stored in a table with uniform sized fields, but instead the data has to be stored as a document having special characteristics. Document stores serve well when the domain model can be split and partitioned across some documents. Document stores should be avoided if the database will have a lot of relations and normalization. They can be used for content management system, blog software etc.

3.2.2.2 Characteristics

Document database systems, are characterized by their schema-free organization of data.

That means:

- Records do not need to have a uniform structure, i.e. different records may have different columns.
- The types of the values of individual columns can be different for each record.
- Columns can have more than one value (arrays).
- Records can have a nested structure.

3.2.3. Column family stores

Column stores in NO SQL are actually hybrid row/column store unlike pure relational column databases. Although it shares the concept of column-by-column storage of columnar databases and columnar extensions to row-based databases, column stores do not store data in tables but store the data in massively distributed architectures. In column stores, each key is associated with one or more attributes (columns). A Column store stores its data in such a manner that it can be aggregated rapidly with less I/O activity. It offers high scalability in data storage. The data which is stored in the database is based on the sort order of the column family. Column oriented databases are suitable for data mining and analytic applications, where the storage method is ideal for the common operations performed on the data.

The following concepts are critical to understand how column databases work:

- Column family
- Super columns
- Column

Columns and super columns in a column database are spare, meaning that they take exactly 0 bytes if they don't have a value in them. Column families are the nearest thing that we have for a table, since they are about the only thing that you need to define upfront. Unlike a table, however, the only thing that you define in a column family is the name and the key sort options (there is no schema).

A Column family is how the data is stored on the disk. All the data in a single column family will sit in the same file. A column family can contain super columns or columns.

A super column is a dictionary, it is a column that contains other columns (but not other super columns).

A column is a tuple of name, value and timestamp.

3.2.4. Graph databases

3.2.4.1 What are Graph databases?

Graph databases are databases which store data in the form of a graph. The graph consists of nodes and edges, where nodes act as the objects and edges act as the relationship between the objects. The graph also consists of properties related to nodes. It uses a technique called index free adjacency meaning every node consists of a direct pointer which points to the adjacent node. Millions of records can be traversed using this technique. In a graph databases, the main emphasis is on the connection between data. Graph databases provides schema less and efficient storage of semi structured data. The queries are expressed as traversals, thus making graph databases faster than relational databases. It is easy to scale and whiteboard friendly. Graph databases are ACID compliant and offer rollback support. Graph databases can be used for a variety of applications like social networking applications, recommendation software, bioinformatics, content management, security and access control, network and cloud management etc. Graph databases are difficult to cluster.

3.2.4.1 Graph databases properties

- **Performance**: data volume will definitely increase in the future, but what's going to increase at an even faster clip is the connections (or relationships) between data. With traditional databases, relationship queries will come to a grinding halt as the number and depth of relationships increase. In contrast, graph database performance stays constant even as data grows year over year.

- **Flexibility**: With graph databases, IT and data architect teams move at the speed of business because the structure and schema of a graph model flex as solutions and industry change. Enterprise's teams doesn't have to exhaustively model their domain ahead of time; instead, they can add to the existing structure without endangering current functionality.

- **Agility**: Developing with graph databases aligns perfectly with today's agile, test-driven development practices, allowing graph-database-backed application to evolve with changing business requirements.

4. Related Works

The need to handle increasingly larger data volumes is one factor driving the adoption of a new class of non-relational "NoSQL" databases: Big Data and especially NoSQL Databases are emerging technologies and several solutions have been proposed to support migration from relational to NoSQL databases. Some of them focus on schema and data migration algorithms and transformation technics. Other approaches focus on queries transformation. Many of other proposals focus on mapping, algorithms and approaches. On the other hand, some approaches have been proposed to the general problem of database translation between different data models but, to the best of our knowledge, all of them are commercial and there is no free tool that offers a complete solution with an approach, algorithm and a tool for migration.

5. Adopted Architecture

We have developed our tool in a Java environment. Experiments were conducted on MOOC database which is a MySQL database and we opted for the Neo4j which is an open-source NoSQL graph database implemented in Java and Scala.

5.1. Neo4j [5]

Neo4j is used today by hundreds of thousands of companies and organizations in almost all industries. Use cases include matchmaking, network management, software analytics, scientific research, routing, organizational and project management, recommendations, social networks, and more. It implements the Property Graph Model efficiently down to storage level. As opposed to graph processing or in-memory libraries, Neo4j provides full database characteristics including ACID transaction compliance, cluster support, and runtime failover, making it suitable to use graph data in production scenarios.
The property graph contains connected entities (the nodes) which can hold any number of attributes (key-value-pairs). Nodes can be tagged with labels representing their different roles in your domain. In addition to contextualizing node and relationship properties, labels may also serve to attach metadata—index or constraint information—to certain nodes.
Relationships provide directed, named semantically relevant connections between two node-entities. A relationship always has a direction, a type, a start node, and an end node. Like nodes, relationships can have any properties. In most cases, relationships have quantitative properties, such as weights, costs, distances, ratings, time intervals, or strengths. As relationships are stored efficiently, two nodes can share any number or type of relationships without sacrificing performance.
Neo4j implements the Property Graph Model efficiently down to the storage level. As opposed to graph

processing or in-memory libraries, Neo4j provides full database characteristics including ACID transaction compliance, cluster support, and runtime failover, making it suitable to use graph data in production scenarios.

Some particular features make Neo4j very popular among users, developers, and DBAs:

- Materializing of relationships at creation time, resulting in no penalties for complex runtime queries
- Constant time traversals for relationships in the graph both in depth and in breadth due to efficient representation of nodes and relationships
- All relationships in Neo4j are equally important and fast, making it possible to materialize and use new relationships later on to "shortcut" and speed up the domain data when new needs arise
- Compact storage and memory caching for graphs, resulting in efficient scale-up and billions of nodes in one database on moderate hardware
- Written on top of the JVM

5.2. MOOC [6]

MOOC (Massive Open Online Courses) is a new paradigm of education for anyone, anywhere, anytime. It came up with numerous opportunities both for students as well as teachers.

• MOOC creates the opportunity for sharing ideas & knowledge and also helps improving lifelong learning skills by providing easy access to global resources.

• It improves cross cultural relationships which lead to collaboration between institution educators and learners locally and internationally.

• It gives an idea where I stand in the course in the current world as large number of students all over the globe would have registered for the same course on the same common platform and participate in the activities and discussion in the study group.

• MOOC enhances active learning. Research shows that students learn more through active learning (i.e. when they have assignments or discussion on an issue) rather than through listening to lectures.

MOOCs offer the entire learning community equal rights to education. People can choose from the varied range of programs and enroll for the course of their choice from the convenience of their homes without spending a dime. Although students study independently in these courses, they at the same time collaborate with their peers from different parts of the world.
MOOCs are a boon for people who have faced obstacles in pursuing education due to lack of funds, no proper opportunities, un-accessible geographical location, etc.

Pedagogically, MOOCs are designed to be extremely interactive. It uses all the interactive media available on the internet to engage students. The various tools used are blogs, videos, podcasts and forums that are embedded into the programs seamlessly. These tools collaborate learners and help them solve real world problems rather than discussing hypothetical material. Real discussion of ideas, theories and concepts are an integral part of a MOOC and are used for peer review and assessment.

6. Data Conversion

This section describes our method for converting a relational database into a graph database, for the sake of simplicity, we consider an intermediate step in which we map the components of a relational database to a graph database component. The joins represented by foreign key relationships will be a factor, but most relationships in a relational database are encoded into table schemas. Each cell in a table becomes an edge, then that edge connects to its column name and to one or more of the other fields of its row. Then, the graph model is automatically generated.

The algorithm could be as (for each table):

- Extract table structure
- Read the table data and find distinct values
- For each row in a table create edges from the value for the key in the table to other non-key values and label the edge with the attribute names. If the table has composite key then create a hyper edge to connect the key domain values to other non-key domain values.

7. Experimental results

7.1. Applying the approach on the MOOC schema

We start by simplifying the MOOC database schema to make it more readable and make it as an ER diagram.

Therefore, entities are displayed as rectangles containing the name of each entity, undirected arrows represent the

relationships between these entities cut each with a diamond containing the non-relation (red dotted square) and cardinality top and bottom of the diamond and attributes of each entity is represented by a small black circle for primary keys and as the number is many other attributes, it is useless to the presented.

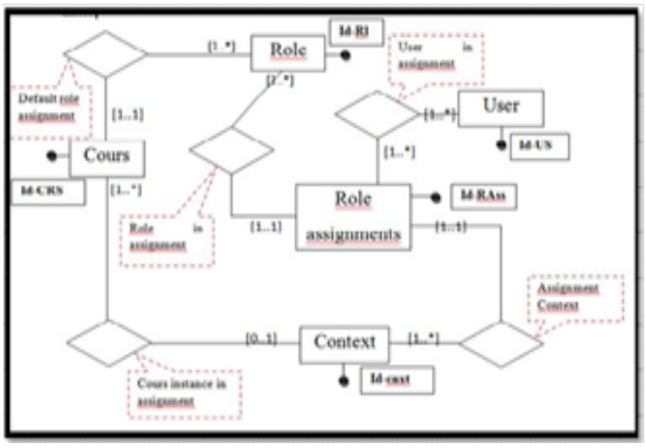


Fig. 1 ER Diagram

7.2. Tool print screens

Our tool produces a graph database starting from a MySQL database (MOOC database in the example below). As shown in fig.2 we begin by choosing the source database.

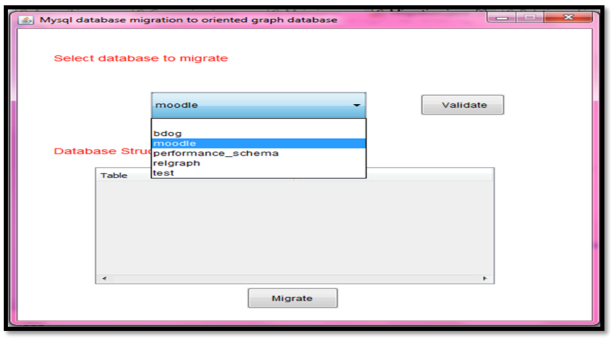


Fig. 2 Selection of the relational database

After choosing the source database which is a relational MySQL database, all tables are shown with their column list Fig.3

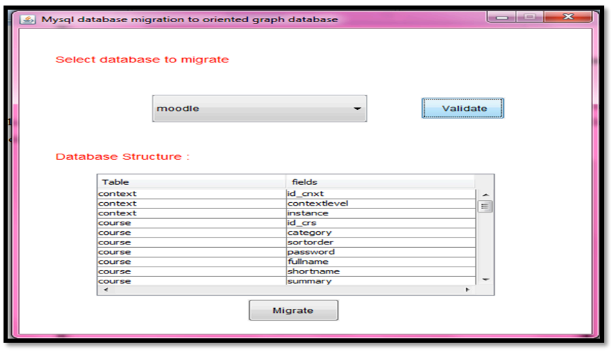
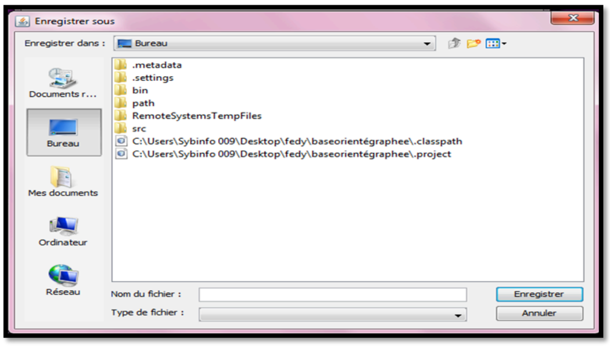

Mysql database migration to oriented graph database
Select database to migrate
moodle
Validate
Database Structure :
Migrate

Table	fields
context	id_cnxt
context	contextlevel
context	instance
course	id_crs
course	category
course	sortorder
course	password
course	fullname
course	shortname
course	summary

Fig. 3Tables and columns

The figure above shows the interface from which the user will choose the storage path the new database that will be generated.



.metadata
.settings
bin
path
RemoteSystemsTempFiles
src
C:\Users\Sybinfo 009\Desktop\fedy\baseorientégraphee\.classpath
C:\Users\Sybinfo 009\Desktop\fedy\baseorientégraphee\.project

Nom du fichier :
Type de fichier :

Fig. 4 Choosing path

Now, our tool deals with the algorithm mentioned earlier and generate a graph database, as an example we choose the database shown in fig. 5

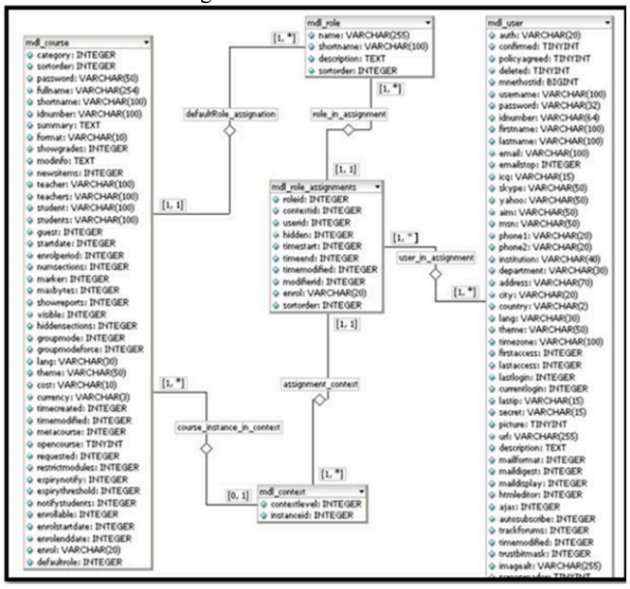
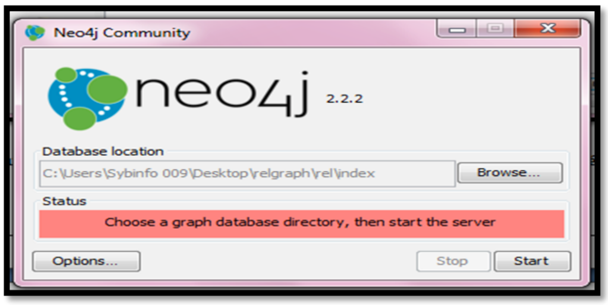


Fig. 5 MOOC database schema

To create a graph database from the generated file we use Neo4j

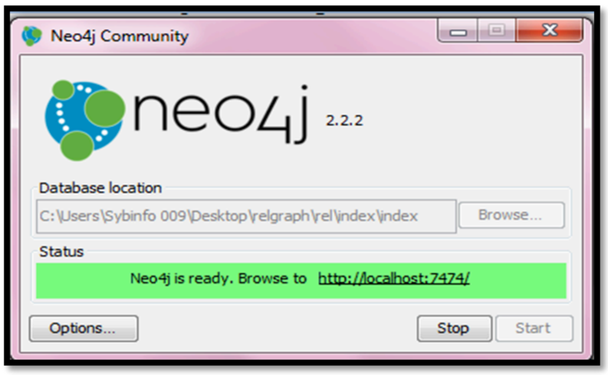


C:\Users\Sybinfo 009\Desktop\relgraph\rel\index

Choose a graph database directory, then start the server

Fig. 6 Browsing the generated file

Once we choose the file, we click on "Start", and then, the graph database will be created fig .7



Neo4j Community

neo4j 2.2.2

Database location

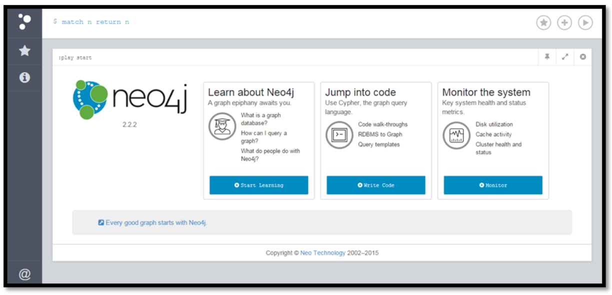
C:\Users\Sybinfo 009\Desktop\relgraph\rel\index\index

Status

Neo4j is ready. Browse to http://localhost:7474/

Fig. 7 Graph database creation

to view the result we'll use a simple CYPHER query **'match n return n'**



```
$ match n return n
```

```
:play start
```

neo4j 2.2.2

Learn about Neo4j
A graph epiphany awaits you.
What is a graph database?
How can I query a graph?
What do people do with Neo4j?

Jump into code
Use Cypher, the graph query language.
Code walk-throughs
RDBMS to Graph
Query templates

Monitor the system
Key system health and status metrics.
Disk utilization
Cache activity
Cluster health and status

Every good graph starts with Neo4j.

Copyright © Neo Technology 2002–2015

Fig. 8 Neo4j Interface

The result of the conversion is a graph database in which the records of the entity "Article" are represented as blue node shown in fig.4, green ones are the records of the entity "commentaire", the yellow nodes represents the records of the entity "image" and the two pink nodes represents two records in the entity "catégorie".

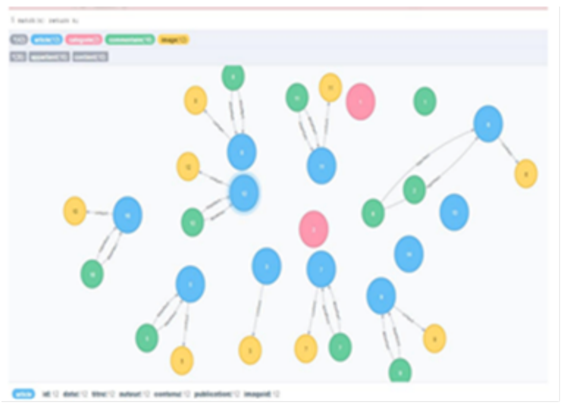


Fig. 9 Results of the conversion

Conclusion

In this document we have presented an approach to migrate automatically schema and data from relational to graph databases. In the futures works we intend to refine the technique proposed in this paper to obtain a more compact target database.

References

[1] Mathieu Roger, synthèse d'étude et projets d'intergiciels: basesNoSQL.

[2] Big Data Explained, retrieved 10 15, 2015, from : www.mongodb.com/big-data-explained..

[3] Learn About NoSQL Databases | NoSQL vs SQL, Benefits | DataStax. Retrieved March 16, 2015, from datastax: www.datastax.com/nosql-databases.

[4] Nayak A, Poriya A, Poojary D. Type of NOSQL Databases and its Comparison with Relational Databases. International Journal of Applied Information Systems (IJAIS 2013).

[5] what_is_neo4j. (n.d.). Retrieved 09 28, 2015, from neo4j.com:neo4j.com/developper/graph-database/#_what_is_neo4j.

[6] Michael Gaebel, "MOOC: Massive Open Online Courses – January 2014", EUA Occasional Papers.

[7] Sadalage, P. (2014, Oct 3). Retrieved 09 28, 2015, from thoughtworks: www.thoughtworks.com/insights/nosql-databases-overview.

[8] cipher-query-language. (n.d.). Retrieved 09 28, 2015, from neo4j.com:neo4j.com/developper/cipher-query-language.

[9] Atzeni, P. Cappellari, R. Torlone, P. A. Bernstein, and G. Gianforme. Model-independent schema translation. VLDB J., 17(6):1347-1370, 2008. https://doi.org/10.1007/s00778-008-0105-2

[10] J. Ricardo, B. Cabral, P. Carreiro, M. Vieira and J. Bernardino. Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data 2015

[11] Jacques B, Reaping the benefits of big data in telecom. Journal of Big Data2016