Normalized NoSQL Graph Data Warehouse

Amal Sellami[1], Ahlem Nabli[1,2], Faiez Gargouri[1]

[1] University of Sfax, MIRACL Laboratory,
Tunisia

[2] Al-Baha University,
Kingdom of Saudi Arabia

sellami.amal91@gmail.com, ahlem.nabli@fss.usf.tn,
faiez.gargouri@isims.usf.tn

Abstract. In the Big Data warehouse context, a graph-oriented NoSQL database system is considered as the storage model which is highly adapted to data warehouses and online analysis. Indeed, the use of NoSQL models allows data scalability and the graph store offers more flexibility when storing and managing massive data. We propose, in this paper, an approach to create a Graph-oriented Data warehouse by transforming Dimensional Fact Model into Graph Dimensional Model. Then, we implement the Normalized Graph Dimensional Model using java routines in Talend Data Integration tool (TOS). The resulting warehouse was evaluated in term of "Read Request Latency" using LDBC-SNB benchmark.

Keywords: Big data, NoSQL, graph databases, data warehouse, normalized transformation rules, extract transform and load.

1 Introduction

During the last decade the explosion of social media has led to the generation of massive volumes of user-generated data and consequently given birth to a novel area of research, namely social network Data Warehousing. This emphasizes how to extend classical data warehouse (DW) methodologies in order to deal with new features of social network data, such as volume, dynamicity and heterogeneity.

A data warehouse is a database for online analytical processing (OLAP) to support decision-making. It is often implemented in the relational database management system (RDBMS). Intuitively, a well-designed DW requires a well-planned logical design all updates and versions of a DW lead to a revision of the logical design. Generally, the mapping from the conceptual to the logical model is made according to three approaches: ROLAP (Relational-OLAP), MOLAP (Multidimensional-OLAP) and HOLAP (Hybrid-OLAP). However, all these models are inadequate when dealing with large amount of data which needs scalable and flexible systems. However, in a constantly connected world, data sources produce increasingly massive data, namely big data.

Traditional relational storage models have shown their limitations in terms of storing and managing big data. Indeed, major players of the web such as Yahoo, Google, Facebook, Twitter and LinkedIn were the first to point out the limitations of the relational model. They found that relational DBMSs are no longer adapted when dealing with an enormous amount of data in the context of distributed environments. Usually, classical DW and On Line Analytical Processing (OLAP) are comprised of a set of concepts like: facts, dimensions, measures and dimension hierarchies, those are used for structured schema representations. However, in case of web-scale applications, many of the dimensional information may not be available in regular structure. Consequently, decision makers are increasingly using NoSQL databases to implement their business solutions. Indeed, as NoSQL database offer great flexibility, they can improve the classic solution based on data warehouses (DW). In the recent years, many web applications are moving towards the use of data in the form of graphs.

For example, social media and the emergence of Facebook, LinkedIn and Twitter have accelerated the emergence of the NoSQL database and in particular graph-oriented databases that represent the basic format with which data in these media is stored. However, some new NoSQL (not-only-SQL) Database Management Systems (DBMSs) have been recently proved to be effective Business Intelligence solutions. They have proven some clear advantages with respect to relational database management systems. Nowadays, the research attention has moved towards the use of these systems for storing "big" data and analyzing it. Different families of NoSQL DBMSs exist: Key-value, Column, Document and Graph. A Key-value database is a collection of data without a schema and organized as a collection of key-value pairs. Data is accessed using the key and its value represents data. A Column database represents data with tables where each row can present different attributes (different columns). A Document database stores information as documents having a complex structure. A Graph database is suited for applications in which there are more interconnections between the data like social networks.

In this paper, we focus on one class of NoSQL stores, namely graph-oriented systems. Graph-oriented systems are used for managing highly connected data and perform complex queries over it. Not only data values but also graph structures are involved in queries. Specifying a pattern and a set of starting points, it is possible to reach an excellent performance for local reads by, first, traversing the graph, then collecting and aggregating information from nodes and edges. Graph-oriented databases are based upon graph theory (set of nodes, edges, and properties).

We recall that data warehousing relies mostly on multidimensional data modeling which is a conceptual model that uses facts to model an analysis subject and dimensions for analysis axes. This conceptual model must then be converted in a graph-oriented logical model. Mapping the multidimensional model to relational databases is quite straightforward, but until now there is no work that considers the direct mapping from the multidimensional conceptual model to NoSQL logical models. The objective of this paper is to model the

normalized logical model of graph data warehouse using java routines in TOS. Then we evaluated our resulting warehouse in term of "Read Request Latency".

This paper is organized as follows. Section 2 represents a literature review. Section 3 introduces an overview of our approach. Section 4 presents the input data source LDBC-SNB. Section 5 discloses our proposal for the data warehouse schem design. Section 6 presents the graph logical model and the transformation rules. Section 7 addresses the creation of Normalized GDM with TOS. Section 8 evaluates the created Normalized Grah Dimensional Model based on set of queries. Section 9 concludes this paper by giving some future research directions.

2 Literature Survey

In the most of existing studies, three variant of transformation approch are proposed (i) an approach that transforms a data warehousing concepts into relational logical model or (ii) an approach that transforms relational data model into NoSQL logical model; (iii) an approach that transfsorms a conceptual model into specific NoSQL DB.

Transformation of data warehousing concepts into relational logical model. Multidimensional databases are mostly implemented using RDBMS technologies. Mapping rules are used to convert structures of the conceptual level (facts, dimensions and hierarchies) into a logical model based on relations. Moreover, many researchers have focused on the implementation of optimization methods based on pre-computed aggregates (also called materialized views, or OLAP cuboids). However, R-OLAP implementations suffer from scaling up to very large data volumes (i.e. "Big Data"). According to Gartner, Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making and process automation. Big Data have led data warehouses towards to distributed environments to store and analyze the large amount of data. Research is currently under way for new solutions such as using NoSQL systems is detailled in [1].

Transformation of a relational data base into NoSQL logical model. Some studies have presented approaches that transform relational DB into NoSQL DB. In the literature, a number of researchers have recognized the deficiencies of the traditional ROLAP data storage and have proposed approaches for the migration from relational databases to NoSQL ones. For example, in [2] two proposals are defined which allow big data warehouses to be implemented under the column oriented NoSQL model. The first one (normalized approach) uses different tables for storing fact and dimension at physical level which requires to achieve the join between tables when aggregation is performed. The second one (denormalized approach) stores the fact and dimensions into one table, which allows to avoid performing join between tables. Furthermore, authors in [3] propose rules allowing the storage of a time dimension in HBase table. In [4], authors propose a set of transformation rules for translating a relational model to column-oriented model via HBase. In [5] an algorithm is introduced

for mapping a relational schema to a NoSQL schema in MongoDB [6], as document oriented NoSQL database. In [7], authors propose a method for transforming object-relational database to NoSQL databases, more especially to the document-oriented databases.

Transformation of a Conceptual Model into NoSQL DB. Few works have focused on the transformation of the multidimensional conceptual model to NoSQL logical one. In [8] authors tried to define logical models for NoSQL data stores (oriented columns and oriented documents). They proposed a set of rules to map star schema into two NoSQL models: column-oriented (using HBase) and document-oriented (using MongoDB). In [9] , authors have proposed a transformation rules that ensure the successful translation from conceptual DW schema to two logical NoSQL models (column-oriented and document-oriented). They also proposed two possible transformations namely: simple and hierarchical transformations. The first one stores the fact and dimensions into one column-family/collection. The second transformation uses different column-families/collections for storing fact and dimensions while explaining hierarchies. In [10] authors focused on simplifying the heterogeneous data querying in the graph-oriented NoSQL systems. In [11] authors give a solution to perform join between tables and performing aggregates from data warehouses implemented according the normalized approach. It consists in the integrating of software and tools such as Hive and Kylin in the ecosystem used for implementing the data warehouse, and use their cube building operators. A recent work, [12] proposed a data storage models for Graph cubes by introducing a document oriented model and a column oriented model for storing a graph cube data and implementing the roll up operation over the MongoDB document-oriented database and Cassandra Column-oriented database. Authors in [13] have proposed already two types of transformation from the multidimensional model to the graph-oriented model. An other approach proposed in [14] propose an approach to create a Graph-oriented Data warehouse by transforming Dimensional Fact Model into Graph Dimensional Model (Denormalized Transformation).

Table 1 summarizes our literature review, based on following four criteria.
C1: Describe the tranformation type of data base uses. (i) from the relational model to NoSQL; (ii) from the conceptual multidemnsional model to NoSQL.
C2: Describe the name of NoSQL database used.
C3: Describe the proposed set of rules for the transformation.
C4: Describe the experimentation used to evaluate his works.

To conclude on the literature review, the majority of approaches propose to transform and create the data warehouse under two NoSQL models (Column and Document oriented NoSQL model). There is no approach that directly transform a data warehouse multidimensional conceptual model into a Graph logical model in order to create a data warehouse using graph data base. The major interesting advantage of the graph oriented model is related to the ability for supporting complex queries without using joins. For that, we propose a new approach to create data warehouse under graph oriented NoSQL data base.

Table 1. Summary of the literature review.

Works	C1	C2	C3	C4
(Stonebraker 2012)	NoSQL	-	-	-
(Rocha L et al., 2015)	Relational to NoSQL	(Column-oriented)	(Simple, Hierarchical)	-
(Li et all., 2010)	Relational to NoSQL	(Column-oriented)	(Simple)	-
(Vajk T et all., 2013)	Relational to NoSQL	(Column-oriented)	(Simple)	-
(Dehdouh et all., 2014)	Relational to NoSQL	(Column-oriented)	(Simple)	✓
(Aicha A et all., 2020)	Relational to NoSQL	(Document-oriented)	-	-
(Chevalier et all., 2015) (a)	Conceptual to NoSQL	(Column-oriented, Document-oriented)	(Simple)	✓
(Chevalier et all., 2015) (b)	Conceptual to NoSQL	(Column-oriented)	(Simple,Hierarchical)	✓
(Dehdouh et all., 2015)	Conceptual to NoSQL	(Column-oriented)	(Simple)	✓
(Yangui et all., 2016)	Conceptual to NoSQL	(Column-oriented, Document-oriented)	(Simple, Hierarchical)	✓
(Elmalki et all., 2018)	Conceptual to NoSQL	(Graph-oriented)	-	-
(Challal et all., 2019)	Conceptual to NoSQL	(Column-oriented,Document-oriented)	(Simple,Hierarchical)	✓
(Sellami et all., 2018)	Conceptual to NoSQL	(Graph-oriented)	(Simple)	-
(Sellami et all., 2020)	Conceptual to NoSQL	(Graph-oriented)	(Simple)	✓

3 Graph NoSQL Warehousing: Approach Overview

In this section, we describe our new approach to design and create data warehouse building under graph-oriented system. This approach is composed of five phases as shown in Fig.1.

Conceptual phase. The conceptual model is designed based on a set of rules from Benchmark LDBC-SNB as data source.

Logical phase. The second phase ensure the transformation of the Conceptual model of DW into the graph-oriented model based on set of rules . We distingush two logical model(Normalized and Denormalized) based on the rules used to transform dimensions.

ETL phase. In the third phase, we are interested on the identification and the implementation of ETL operations. The ETL operations are implemented under TOS. The result of this phase is two logical models (Normalized and Denormalized) based on the graph paradigm.

Comparative Study. This phase has as input the two logical models and perform a comparative study in order to choose the best logical model based on two metrics: Write-Request-Latency (WRL) and Read-Request-Latency (RRL). WRL measures the loading time for a single write, and RRL measures the response time of a query.

Reporting Queries. In this phase, we propose to use Cypher graph query language to create and analysis a report of the graph oriented data warehouse. The visualization of the query is done using powerBI.

4 LDBC'S Social Network

As input data source we used the Linked Data Benchmark Council Social Network Benchmark. The LDBC SNB is generated using data generator (DATAGEN) evolved from the S3G2 generator.The LDBC SNB aims at being a comprehensive benchmark by setting the rules for the evaluation of graph-like data management technologies. LDBC SNB is designed to be a plausible look-alike of all the aspects of operating a social network site, as one of the most representative

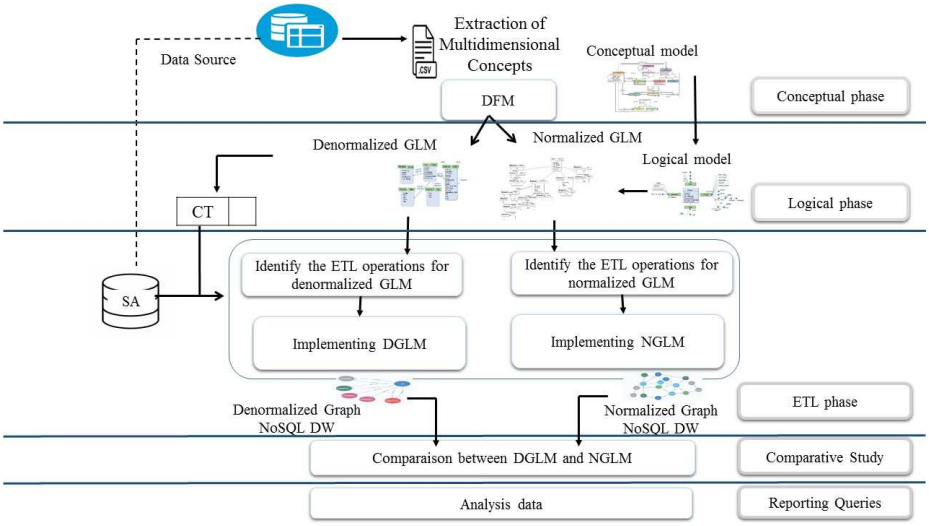

Fig. 1. Approach overview.

and relevant use cases of modern graph-like applications. Its schema has 11 entities connected by 20 relations, with attributes of different types and values, making for a rich benchmark dataset.

A detailed description of the schema is found at [15, 16]. Fig.2 shows the LDBC data schema in UML. The schema defines the structure of the data used in the benchmark in terms of entities and their relations. Data represents a snapshot of the activity of a social network during a period of time. Data includes entities such as Persons, Organisations, and Places. The schema also models the way persons interact, by means of the friendship relations established with other persons, and the sharing of content such as messages (both textual and images), replies to messages and likes to messages. People form groups to talk about specific topics, which are represented as tags.

5 Data Warehouse Schema Design

We propose, in this section, the design of the data warehouse schema based on a set of rules proposed in [17]. These rules applied on the LDBC-SNB Benchmark are used to identify the multidimensional concepts precisely of fact, measures,dimensions and hierarchies.

5.1 Determination of Fact and Measures

An analyzed subject represented by the concept of fact. Each fact characterized by one or more measure representing the indicators analyzed. To extract the fact and its measures, we apply the following rules:

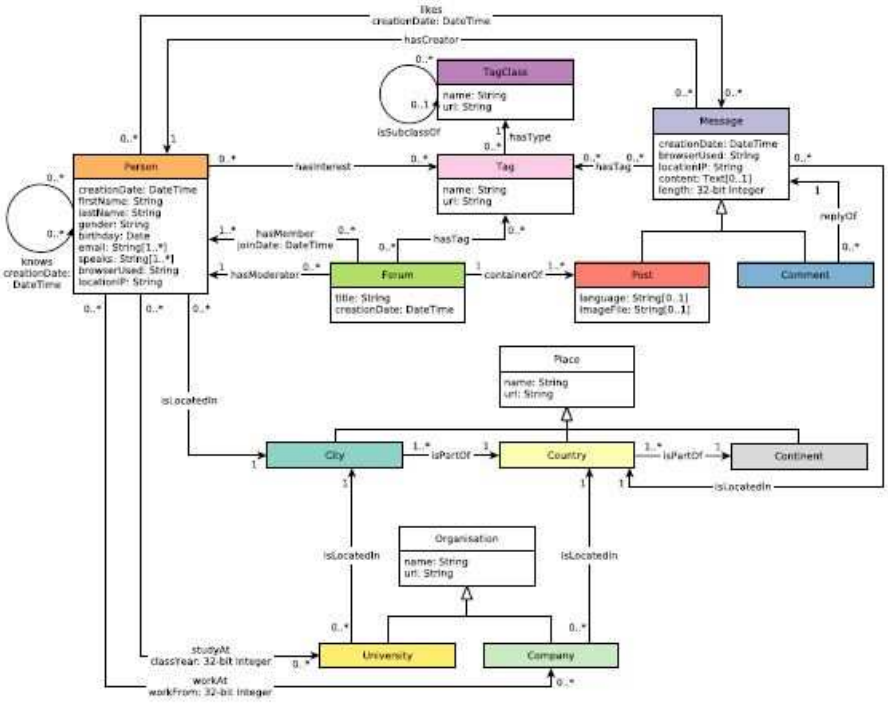

Fig. 2. The LDBC-SNB data schema.

– **Identification of a fact.** We are interested in our work in the analysis of the forum. So we obtain the fact forum.
– **Identification of measures.** The measures are generally numeric and correspond to the "how much" or "how many" aspects of a question. Each measure in a fact should have a default aggregation (or derivation) rule. The data retreived from LDBC-SNB allows us to obtain the following measures: Number of Post, Number of Member, Number of Moderator, Number of Tag, Number of comment and Number of like. Table 2 presents the name of the class / relationship / attributes in our benchmark, the name of the determined measure and its description.

Table 2. Forum Measures.

Class / RlationShip / Atributes Name	Measures Name	Description
Post / Container of	nbPost	The total number of a Post in forum
Person / Has member	nbMembre	The total number of member in forum
Person / Has moderator	nbModerator	The total number of forum moderator
Tag / Has tag	nbTag	The total number of Tag representing in the forum's
Message / Reply of Comment	nbComment	The total number of comment in forum

5.2 Determination of Dimensions

The extraction of dimensions is based on a type of object called basic object. A dimension represents a single set of objects or events in the real world. Dimensions are the qualifiers that make the measures of the fact table meaningful, because they answer the what, when, and where aspects of a question. Based on these questions, we obtain four dimensions for forum which are: Person, Date, Message and Tag (Table 3).

Table 3. Determination of dimensions from LDBC-SNB.

Questions	Basic Object
What contains the forum?	Message
Who moderats the forum?	Person
When the forum is published?	Date
Which words are used to describe the forum?	Tag

For each determined dimension, we detail in the following subsections its parameters. Each attribute or class not chosen as a measure can be an attribute for a dimension specially the categorical attributes.

Determination Date Dimension Attributes. The date is an information that is saved in each record of the data source. The date dimension is a mandatory dimension for analysis and interrogation. The definition of the granularity of the date dimension is based on the need of decision makers, according to which granularity leads its analysis.Table 4 shows the attributes composing the date dimension. The dimensional elements for the Date dimension are day, month and year. Day has a roll-up hierarchical relationship with month which has a roll-up hierarchical relationship with year. Day has an attribute of id date. Fig.3 (a) illustrates the date dimension according to the DFM formalism.

Table 4. Attributes of Date dimension.

Atributes Name	Description	Type
IdDate	The identifier of the date	Identifier
Day	Day of date	Level 2 parameter
Month	Month of date	Level 3 parameter
Year	Year of date	Level 4 parameter

Determination Person Dimension Attributes. The person is an abstract entity that represents a person. It contains various information about the person as well as network related information. The attributes of Person class are: id, firstName, lastName, gender, birthday, email, speaks, browserUsed, locationIP and creationDate. As relationships with person we retreived "islocatedin" and "studyat". The relationship "islocatedin" describe a person and their home

located. There is a class place with attributes name and url. City, country and continent are a sub-class of a place. The relationship "studyat" describe the organisation of the person studied. The class organization have the attributes name, url and have to sub-class university, company.

So, we can describe the Person dimension (D_{Person}) with parameters ID_{Person} along with the weak attributes ($First_{name}$, $Last_{name}$, Birthdate, E_{mail}, $Creation_{Date}$), organized using four hierarchies Hgendre, Hspeaks, Hplace and Horganisation. Table 5 illustrates the set of attributes making up the Person dimension. Fig.3 (b) illustrates the person dimension according to the DFM formalism.

Table 5. Attributes of Person dimension.

Atributes Name	Description	Type
IdPerson	The identifier of the person	Identifier
FirstName	The first name of the person	Weak attribute
LastName	The last name of the person	Weak attribute
Birthday	The birthday of the person	Weak attribute
Email	The email of the person	Weak attribute
CreationDate	The date the person joined the social network	Weak attribute
Gender	The gender of the person	Level 2 parameter
Speaks	The set of languages the person speaks	Level 2 parameter
OrganisationName	The name of the organisation	Level 2 parameter
Url	The url of the organisation	Weak attribute
Type	The type of the organisation	Level 3 parameter in the organisation hierarchie
Label	The label of the organisation	Weak attribute
PlaceName	The name of the place	Level 2 parameter
City	The city of the place	Level 3 parameter in the place hierarchie
Country	The country of the place	Level 4 parameter in the place hierarchie
Continent	The continent of the place	Level 5 parameter in the place hierarchie

Determination Message Dimension Attributes. The message is an abstract entity that represents a message created by a person. The attributes of Message entity: creationDate, browserUsed, locationIP, content and length. Post and comment are a sub-class of Message, it is defined as a type of the message.

Posts contain either content or imageFile, always one of them but never both. Table 6 shows all the set of attributes composing the message dimension. Fig.3 (C) illustrates the message dimension according to the DFM formalism.

Table 6. Attributes of Message dimension.

Atributes Name	Description	Type
IdMessage	The identifier of the message	Identifier
CreationDate	The date the message was created	Weak attribute
Content	The content of the message	Weak attribute
Length	The length of the content	Weak attribute
TypeMessage	The type of the message	Level 2 parameter
BrowserUsed	The browserused of sent the message	Level 2 parameter

Determination Tag Dimension Attributes. Tag is used to specify the topics of forums. The attributes of Tag entity: id, name and url. Table 7 shows all the set of attributes composing the Tag dimension. Fig.3 (d) illustrates the Tag dimension according to the DFM formalism.

Table 7. Attributes of Tag dimension.

Atributes Name	Description	Type
ID	The identifier of the tag	Identifier
Name	The name of the tag	Weak attribute
Url	The URL of the tag	Weak attribute
Type	The type of the tagclass	Level 2 parameter

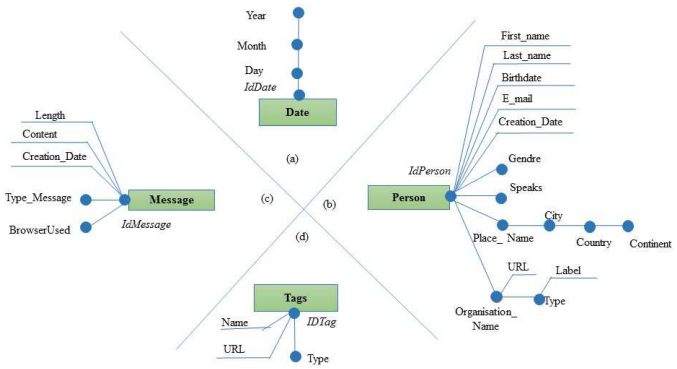

Fig. 3. Dimensions of our LDBC-SNB data warehouse.

From the previous steps, we obtain the conceptual model of data warehouse schema generated from the data source LDBC-SNB (Fig.4).
The mapping from the conceptual to the logical model is made according to three approaches: ROLAP (Relational-OLAP), MOLAP (Multidimensional-OLAP) and HOLAP (Hybrid-OLAP). All these models are inadequate when dealing with large amount of data which need scalable and flexible systems. As an alternative, NoSQL systems begin to grow. In our case we are oriented to use the NoSQL graph-oriented databases.

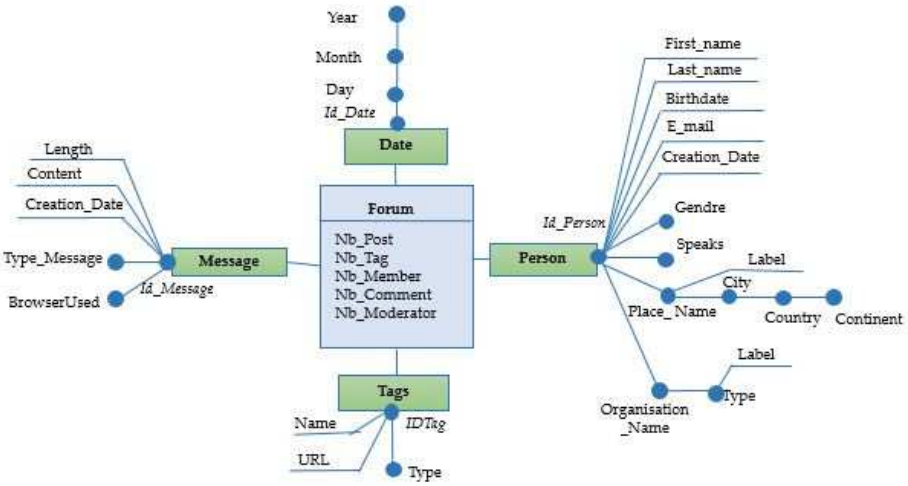

Fig. 4. Multidimensional Conceptual Model: DFM.

6 Graph Logical Model

Graph technology has been the fastest growing category of databases in recent years. In the world where connected data represents a new source for companies, graph technology appears as the obvious option. Therefore, NoSQL Graph-oriented databases are perfectly adapted to voluminous, heterogeneous and massively interconnected data due to its flexible structure capable to representing elegantly correlated and dynamic data.

This kind of database are based upon graph theory. Data is represented as nodes, edges and attributes, which allows the modeling of different interactions between data. Graphs modeling is ubiquitous in most social networks, semantic web and bio science (protein interactions ...) applications.

Graph-oriented systems belong within the "schema less" framework, that consists in writing data without any prior schema restrictions; i.e., each node and each edge have its own set of attributes, thus allowing a wide variety of representations. This flexibility generates heterogeneous data, and makes their interrogation more complex for users, who are compelled to know the different schema of the manipulated data.

It is useful for inter-connected relationship data. The relational database performed better on executing queries when the amount of data is relatively limited. However, as queries became complex, the graph database outperformed the relational one. For the conceptual modeling, an entity-relationship diagram is readily translated into a Property Graph Model, making a conceptual model for graph databases necessary. It helps to understand which entities can be logically connected to which other entities. Graph databases support only binary relation-

ships. On the other hand, graph modeling is much easier than for a relational data model because real world objects are explicit in terms of connections.

The data modeling in NoSQL graph-oriented systems consists in representing the database as a graph. The reason why graph databases are an interesting category of NoSQL is because, contrary to the other approaches, they actually go the way of increased relational modeling, rather than doing away with relations. that is, one to one, one to many, and many to many structures can easily be modeled in a graph-based way. In a way, a graph database is a hyper-relational database, where JOIN tables are replaced by more interesting and semantically meaningful relationships that can be navigated (graph traversal) and/or queried, based on graph pattern matching. The data modeling in NoSQL graph-oriented systems consists in representing the database as a graph.

Formally, we can represent a NoSQL **Graph-oriented database** as G (V, E, P) where:

- V is a set of node that represent the entities,
- $E^G = E_1, \ldots, E_y$ is a set of edges that represent the relation between the nodes,
- P is a set of properties attributed to each component of the graph-oriented database (node/arc). A property is formed by a couple of a key and value pair.

Node. Each node has property and label. Formally, a node, is defined by (id^V, P^V, L^V) where:

- id^V is the identifier of the nodes,
- P^V is a set of properties that describe a node,
- L^V is a set of labels or etiquette attached to the node. In order to express the semantic of the nodes, usually a node can have 0 or more labels written as $L^G = L_1, \ldots, L_q$.

Relation. The relations connecting the nodes can eventually have properties. Formally, a relation is defined by (id^R, Vi^R, Vo^R, T^R, P^R) where:

- id^R is the identifier of the relation,
- Vi^R is the identifier of the incoming node,
- Vo^R is the identifier of the outgoing node,
- T^R is the type of relation that bears the name of the relation,,
- P^R is a set of properties of a relation.

In order to implement the data warehouses within the graph-oriented NoSQL model, we propose two transformations namely DLM (Denormalized Logical Model), and NLA (Normalized Logical Model). Each one differs in terms of the structure and the attribute types used when mapping is performed. In the following we details the two transformations rules. Each transformation load to a graph logical model.

6.1 Transformation rules: Normalized logical model

Recall that a data warehouse schema consists of fact with measures, as well as a set of dimensions with attributes, we map the dimensions according to its attributes and the facts according to its measures. The normalized logical transformation ensures the mapping from the multidimensional model of DW to NOSQL Graph logical model, while explaining hierarchies. In this transformation each fact and dimension are transformed into nodes according the following rules:

Rule 1: Transformation of a fact and its measures to the graph-oriented model.

Fact/Measures transformation. Each fact is transformed into a node with the label of the node takes the type of the concept of the multidimensional model which is 'fact' then we add the name of the fact as a second label at the same node. Each measure is transformed by a property of Fact node.
RF.1. Each fact F $\in F^{MS}$ is transformed into a node, defined by V (id^V, P^V, L^V) where:

- Label l_1 is the type of the multidimensional concept: l_1='Fact' / L^V=$\{l_1\}$,
- Label l_2 is the name of the fact: l_2=N^F / L^V=$L^V \cup \{l_2\}$,
- Each measure $m_i \in M^F$ is represented as a property with p $\leftarrow m_i$/P^V=$P^V \cup$\{p\}.

Rule 2: Transformation of a dimension and its attributes(Strong and Weak) to the graph-oriented model.
Rule.2- Dimension/Parameters transformation. Each dimension is transformed into a node with the label of the node takes the name of the concept of the multidimensional model (in this case is the dimension). Then, we use the name of the dimension as a second label at the same node. After, the identifier is transformed into a property in the node. Finally, any weak attribute associated to the identifier is transformed into a property in the same node. After that, each weak attribute is represented in the form of property.
RD.1. Each name and identifier of a dimension is transformed into a node V (id^V, P^V, L^V) where:

- Label l_1 is the type of the multidimensional concept: l_1='Dimension' / L^V=$\{l_1\}$,
- Label l_2 is the name of the dimension: l_2=N^D / L^V=$L^V \cup \{l_2\}$,
- The identifier a_i modeling by a property p with p $\leftarrow a_i$/P^V=$P^V \cup$\{p\},
- Each weak attribute a_w associated to a_i is transformed into a property p with p $\leftarrow a_w$/P^V=$P^V \cup$\{p\}.

Rule.3- Hierarchies transformation.A hierarchy consists of a set of parameters and a link of precedence between parameters. Each parameter is transformed into a node with the label of the node takes the name of the concept of the multidimensional model (in this case is the parameter). Then, we allow the name of the parameter as a second label at the same node. After that, each weak attribute is represented in the node in the form of property. Finally, each link of precedence is transformed into a relation.

RH.1. Transformation of parameter / Modeling the link of precedence between parameter

RH.1.1 Transformation of parameter

Each parameter a_i is transformed into a node V (id^V, P^V, L^V) where:

- Label l_1 is the type of the multidimensional concept: l_1='Parameter' / L^V=$\{l_1\}$,
- Label l_2 is the name of the parameter: l_2=a_i / L^V=$L^V \cup \{l_2\}$,
- Each weak attribute a_w associated to a_i is transformed into a property p with p $\leftarrow a_w$/P^V=$P^V \cup$\{p\}.

RH.1.2. Transformation of link of precedence between parameter

Each $a_i \rightarrow a_{i-1} \subset H^D$ is transformed into a relation R, defined by (id^R, $V1^R$, $V\emptyset^R$, T^R, P^R) where:

- $V1^R$ is the node represented a_i,
- $V\emptyset^R$ is the node represented a_{i-1},
- The type t_1 is the name of the relation: t_1='Precede'/T^R=$\{t_1\}$.

Rule 4: Transformation of the link between the fact and dimension to the graph-oriented model.

Rule.4- Link fact-dimension transformation. Each link between fact and dimension is represented as a relation having as node source the node modeling the fact and as node destination the node modeling the dimension. The relation has as name 'link fact-dimension'.

RFD.1. Each link fact-dimension is transformed into a relation R, defined by (id^R, $V1^R$, $V\emptyset^R$, T^R, P^R) where:

- $V1^R$ is the node represented the fact,
- $V\emptyset^R$ is the node represented the dimension,
- The type t_1 is the name of the relation: t_1='link fact-dimension'/T^R=$\{t_1\}$.

As output of this transformation is normalized logical model for graph data base and a correponding table with full documentation of all transformation operations. This table will be used in ETL process for modeling and implementing the transformation rules within ETL. As NoSQL DBs are schema-less, this increases the need for extending the existing ETL tool in order to be able to create data warehouse while integrating data. ETL tool should be adapted with the constant changes, to produce and to modify executable code quickly. An example of the correpondence table for the normalized logical model is presented in Table 8.

Table 8. Example of correspondence table for NLM.

Object source	Type	Operation	Target	Type Data
Forum	Fact	Rule1: Fact Transf	Forum	Node
Nb_Tag	Measures	Rule1: Measures Transf	Nb_Tag	Property
Message	Dimension	Rule2: Dimension Transf	Message	Node
Place	Weak attribute	Rule3: Hierarchie Transf	Place	Node

The application of the normalized logical transformation rules on the dimensional fact model of Fig. 4 provides the logical model of a data warehouse using the graphic formalism illustrated by Fig.5.

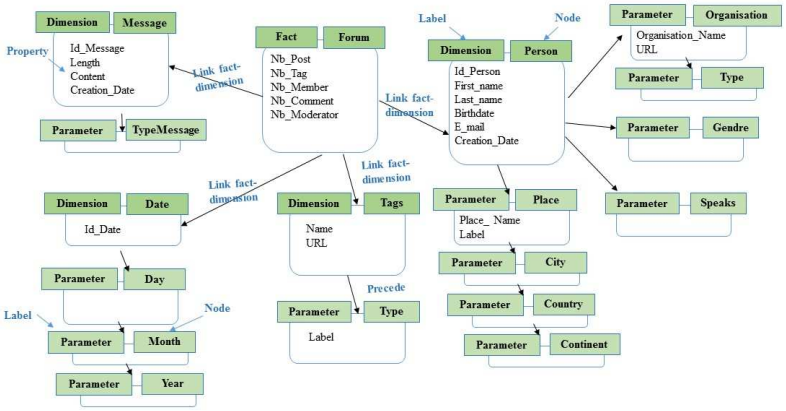

Fig. 5. Normalized Graph Dimensional Model.

6.2 Transformation Rules: Denormalized Logical Model

We recall that, the denormalized Logical transformation ensures the mapping to NoSQL model while highlighting the concepts of the Multidimensional Schema (MS) but without detailing the hierarchies. In this transformation we use 3 rules as follows:

Rule 1: Transformation of a fact and its measures to the graph-oriented model.
Rule.1- Fact/Measures Transformation. Each fact is transformed into a node with the label of the node takes the type of the concept of the multidimensional model which is 'fact' then we add the name of the fact as a second label at the same node. Each measure is transformed by a property of Fact node.

Rule 2: Transformation of a dimension and its attributes (Strong and Weak) to the graph-oriented model.
Rule.2: Dimension/Parameters Transformation.
Each dimension is transformed into a node with the label of the node takes the name of the concept of the multidimensional model (in this case is the dimension). Then, we allow the name of the dimension as a second label at

the same node. After, the identifier is transformed into a property in the node. Finally, any weak attribute associated to the identifier is transformed into a property in the same node. Each parameter is transformed into a property in the node (dimension). After that, each weak attribute is represented in the form of property.

Rule 3: Transformation of the link between the fact and dimension to the graph-oriented model.

Rule.3: Link fact-dimension Transformation.

Each link between fact and dimension is represented as a relation having as node source the node modeling the fact and as node destination the node modeling the dimension. The relation has as name 'link fact-dimension'.

The application of the proposed denormalized Logical transformation rules on the dimensional fact model is illustrated in Fig.6.

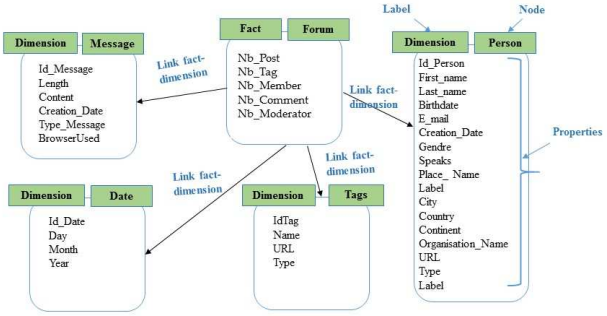

Fig. 6. Denormalized Graph Dimensional Model.

As we previously mentioned, in this level, the correspondence table is generated to keep trace of different transformations. Table 9 presents an excerpt of the generated correspondence table (CT). This table is useful for the ETL process.

Table 9. Example of correspondence table for DLM.

Object source	Type	Operation	Target	Type Data
Forum	Fact	Rule1: Fact Transf	Forum	Node
Nb_Post	Measures	Rule1: Measures Transf	Nb_Tag	Property
Message	Dimension	Rule2: Dimension Transf	Message	Node
IDPerson	Weak attribute	Rule2: Parameter Transf	Place	Property

7 Implementing the Normalized logical transformation Rules: ETL Process

Traditional ETL is a type of data integration from multiple sources (structured and semi-structured data), that follows three steps (extraction, transformation, and loading to a data warehouse or data mart). Its goals are the organization and the storage of data in unified format frequently as a data warehouse.
To load data in the Graph NoSQL DWs, we choose to use the data integration tool "Talend for Big Data". This tool allows extracting data from large and heterogeneous data sources and integrates them into NoSQL database. In the context of our work, data integration is done according to our transformation rules. These rules are implemented using ETL routines in the same tool. The key component of the ETL process is the Job. It is a graphical design, of one or more components connected together such as: tFileInputDelimited (PersonFile, DateFile, ect.), tMap, tNeo4jConnection, tNeo4jRow, tNeo4jOutputRelationship. This components are described as follows:
-tFileInputDelimited reads a given file row by row with simple separated fields. It purpose to open a file and read it row by row to split them up into fields then sends fields as defined in the Schema to the next Job component, via a Row link.
-tNeo4jOutputRelationship receives data from the preceding component, and writes relationships into Neo4j. It is used to output relationship into a Neo4j database.
-tNeo4jOutput receives data from the preceding component, and writes the data into Neo4j. It is used to write data into a Neo4j database, and/or update or delete entries in the database based on the index defined.
-tNeo4jRow is the specific component for this database query. It executes the stated Cypher query onto the specified database. The row suffix means the component implements a flow in the Job design although it doesn't provide output. It depending on the nature of the query, tNeo4jRow acts on the data (although without handling data).
-tMap is one of the core components of Talend Studio and is used very often in Jobs. The tMap component is primarily used for mapping input fields to output fields and transforming the input data in the Expression Builder of the corresponding output column.

For implementing the graph-oriented DW, we use Neo4j. The graph model in Neo4j consists of a Property, only edges can be associated with a type and Edges can be specified as directed or undirected. Neo4j uses the following index

mechanism: a super reference node is connected to all the nodes by a special edge type "REFERENCE". This actually allows to create multiple indexes to distinguish them by different edge types.

All these components are used to create the data warehouse as depicted in Fig.7.

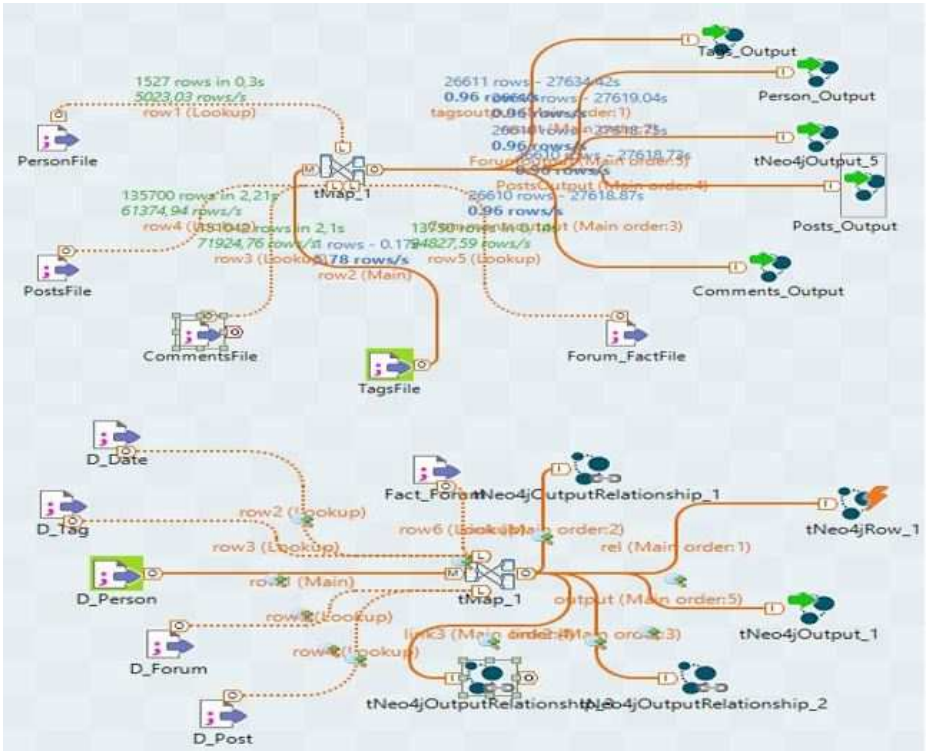

Fig. 7. Creation of Graph DW under TOS.

An example of the number of rows, reading time and loading time (in seconds) of some input files are detailled in Table 10.

The created data warehouse is visualized under Neo4j as presented in Fig.8. As shows in Fig.8, it is composed of 29192 noeuds and 39800 relationShips.

8 Evaluation

Cypher is Neo4j's graph query language that allows users to store and retrieve data from the graph database. The Cypher query language depicts patterns of nodes and relationships and filters those patterns based on labels and properties.

Like SQL, Neo4j CQL has provided some aggregation functions to use in RETURN clause. It is similar to GROUP BY clause in SQL. We can use this

Table 10. Obtained results.

Input File	Number Rows	Reading Time	Loading Time
PersonFile	1527 rows	0,3s	0,8s
TagsFile	16079 rows	0,72s	0,96s
PostFile	135700 rows	0,76s	0,96s
CommentsFile	151042 rows	0,9s	1,2s
ForumFile	13750 rows	0,5s	0,90s
PlaceFile	1495 rows	0,9s	1,2s
DateFile	1095 rows	0,2s	0,8s
OrganisationFile	2258 rows	0,6s	1,2s
Fact_ForumFile	965800 rows	4,98s	9,2s

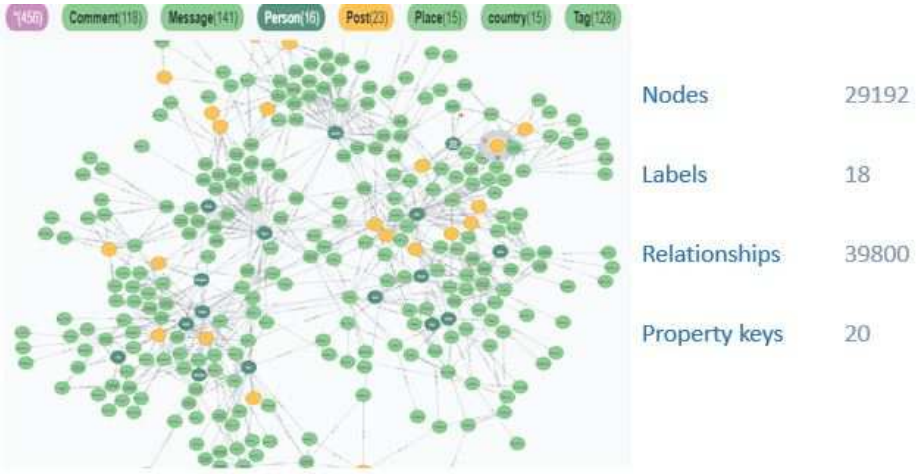

Fig. 8. Normalized Graph NoSQL DW.

RETURN + Aggregation Functions in MATCH command to work on a group of nodes and return some aggregated value.

The aggregate function can take multiple values and can calculate the aggregated values for them. Four levels of pre-aggregates are computed on top of the benchmark generated data. Precisely, at each level we aggregate data respectively on: the combination of 4 dimensions all combinations of 3 dimensions, all combinations of 2 dimensions, all combinations of 1 dimension, 0 dimensions (all data). At each aggregation level, we apply aggregation functions: max, min, sum and count on all dimensions.

In this paper we evaluate the created GDW based on 4 queries described in Table 11.

We measure the efficiency of the implemented NoSQL Graph DW with the metric Read-Request-Latency (RRL). RRL measures the response time of a query. Table 12, summarize the result of graph analysis oriented data warehouse using cypher query language (1 to 4).

Table 11. Query description.

Request	Neo4j Langage
Query 1	MATCH(p:Person) MATCH(Fact:FactForum) MATCH(Mes:Message) RETURN Fact.nbpost ORDER BY p.gender, Mes.browserUsed;
Query 2	MATCH(p:Person) MATCH(Fact:FactForum) MATCH(Mes:Message) RETURN Fact.nbpost ORDER BY p.langue, Mes.browserUsed;
Query 3	MATCH(p:Person) MATCH(Fact:FactForum) MATCH(Mes:Message) MATCH(D:Date) RETURN Fact.nbpost where p.gender='female', D.year='2012' ORDER BY p.gender, Mes.browserUsed, D.month;
Query 4	MATCH(T:Tag) MATCH(Fact:FactForum) MATCH(Mes:Message) RETURN Fact.nbtag ORDER BY T.nametag, Mes.browserUsed;

Table 12. Query analysis based in RRL.

Number Records	Request	RRL(s)
80000	Query 1	0,21
	Query2	0,25
	Query3	0,51
	Query 4	0,32
100000	Query 1	0,29
	Query2	0,33
	Query3	0,64
	Query 4	0,46

The visualization of the query analysis of graph oriented data warehouse using cypher query language is done using powerBI.

Query1: This query gives the number of post by gender and browserUsed. Fig.9 shows the result of Q1 inspired on the Graph DW. *Query2:* This query gives the number of post by langue, browserUsed and year. Fig.10 shows the result of Q2 inspired on the Graph DW. *Query3:* This query gives the number of post by month name and browserUsed, with a clause in the property gender and the year. Fig.11 visualize the result of Q3 inspired on the Graph DW. *Query4:* This query gives the number of moderator by name of the tag and browserUsed.

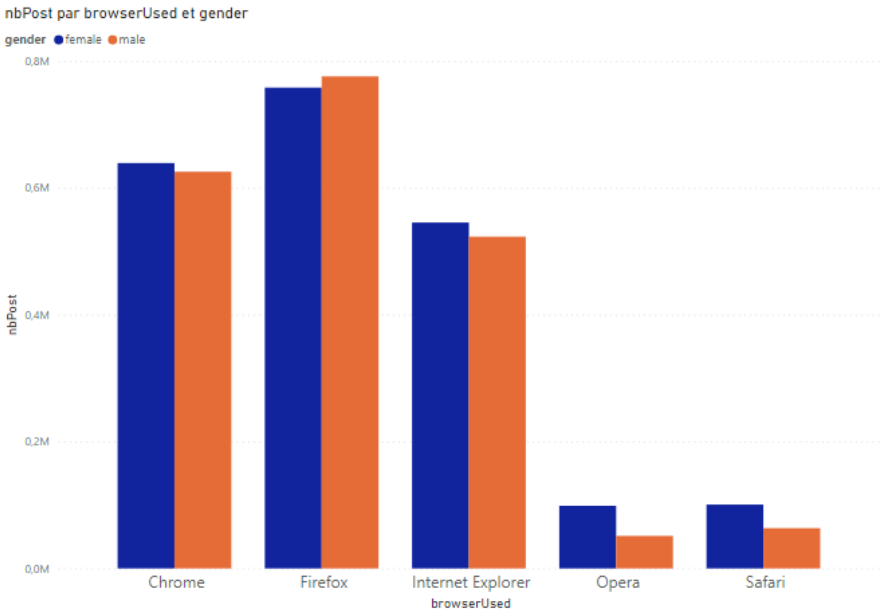

Fig. 9. Number of post by gender and browserUsed.

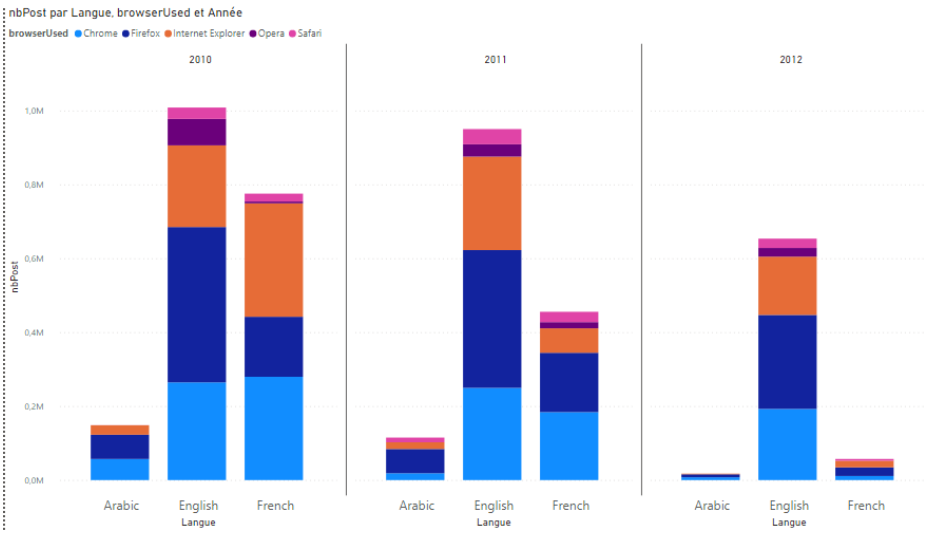

Fig. 10. Number of post by langue, browserUsed and year.

Fig.12 visualize the result of Q4 inspired on the Graph DW.

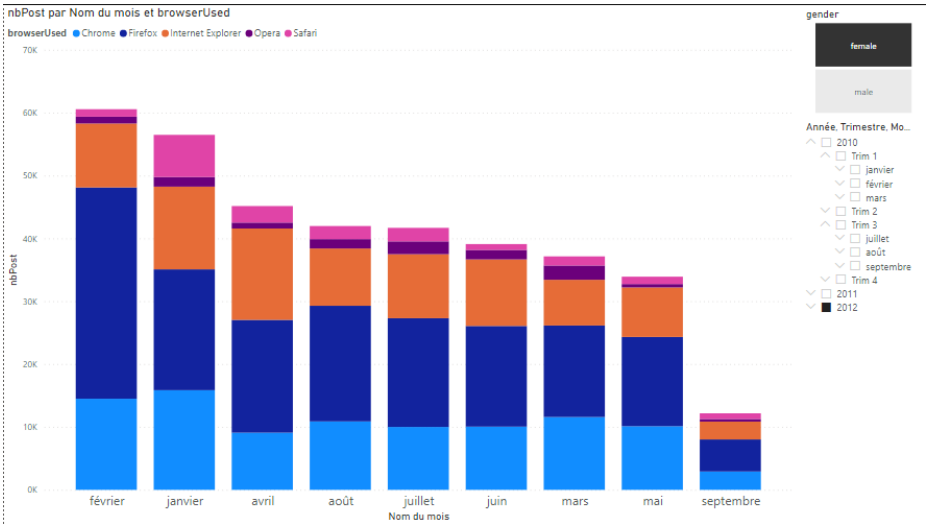

Fig. 11. Number of post by month name and browserUsed, with a clause in the property gender and the year.

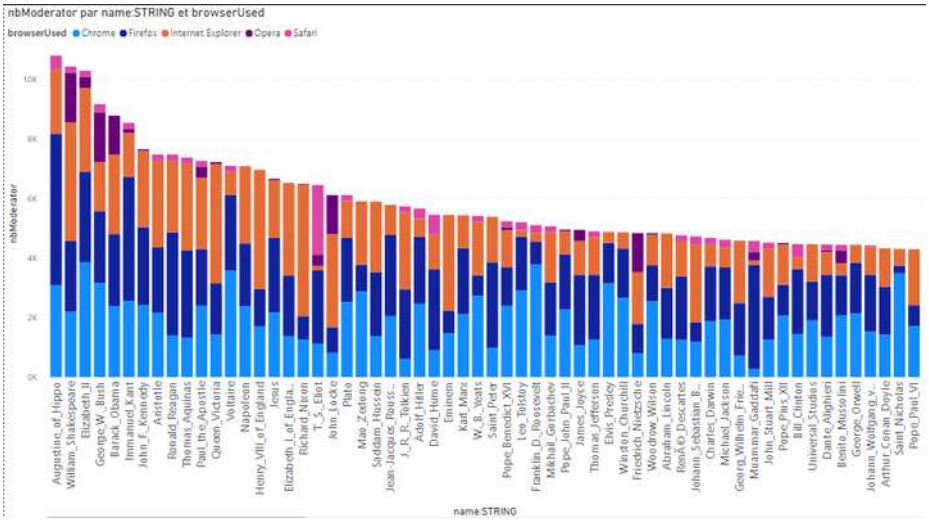

Fig. 12. Number of moderator by name of the tag and browserUsed.

9 Conclusion

As big data continues down its path of growth, a major challenge of the decisional information systems has become how to deal with the explosion of data and its analysis when the data warehouses are implemented.

Consequently, the implementations of data warehouses are oriented towards the new technologies in order to allow more scalability and flexibility for storing and handling data. Since the relational systems are lack of scaling and inefficient of handling big data it is vital to extract transform and loading the data into graph NoSQL data warehouse.

We propose, in this paper, an approach to create a Graph-oriented Data warehouse. We identified two transformations named normalized and denormalized. We have focused on the normalized transformation. Then, we have implemented the Normalized Graph Dimensional Model using java routines in Talend Data Integration tool (TOS).

After that, we evaluated our approach using a set of OLAP queris. As future work, we aim to carry a comparative study in order to choose the best transformation between normalized and denormalized one.

References

1. Stonebraker, M.: New opportunities for new SQL. Commun. ACM 55(11), 10–11 (2012). http://doi.acm.org/10.1145/2366316.2366319.
2. Rocha, L., Vale, F., Cirilo, E., Barbosa, D., Mourão, F.: A framework for migrating relational datasets to NoSQL. Procedia Computer Science 51, 2593–2602 (2015)
3. Li, C.: Transforming relational database into HBase: A case study. In: ICSESS'10. pp. 683–687. IEEE (2010)
4. Vajk, T., Fehér, P., Fekete, K., Charaf, H.: Denormalizing data into schema-free databases. In: CogInfoCom'13, pp. 747–752, IEEE (2013)
5. ONeil, P., ONeil, E., Chen, X., Revilak, S.: The star schema benchmark and augmented fact table indexing. In: Performance Evaluation and Benchmarking, vol. 5895, pp. 237–252. Springer Berlin Heidelberg (2009)
6. Dehdouh, K., Boussaid, O., Bentayeb, F.: Using the column oriented NoSQL model for implementing big data warehouses. In: Proceedings of the 21st International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 469–475 (2015)
7. Aggoune, A., Namoune, M. S.: A Method for Transforming Object-relational to Document-oriented Databases. In: International Conference on Mathematics and Information Technology, Adrar, Algeria (2020)
8. Chevalier, M., El Malki, M., Kopliku, A., Teste, O., Tournier, T.: Implementing Multidimensional Data Warehouses into NoSQL. In: International Conference on Enterprise Information Systems David Harel. First-Order Dynamic Logic. Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, NY (1979). https://doi.org/10.1007/3-540-09237-4.
9. Yangui, R., Nabli, A., Gargouri, F.: Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study. Procedia Computer Science, vol. 96, p. 255-264 (2016)
10. El Malki, M., Ben Hamadou, H., Chevalier, M., Péninou, A., Teste, O.: Querying Heterogeneous Data in Graph Oriented NoSQL Systems. Big Data Analytics and Knowledge Discovery - 20th International Conference, DaWaK (2018)
11. Chavan, V., Phursule, R.: Survey paper on big data. International Journal of Computer Science and Information Technologies, 5(6), 7932–7939 (2014)

12. Challal, Z., Bala, W., Mokeddem, H., Boukhalfa, K., Boussaidy, O., Benkhelifa, E.: Document-oriented versus Column-oriented Data Storage for Social Graph Data Warehouse. (2019)
13. Sellami, A., Nabli, A., Gargouri, F.: Transformation of Data Warehouse Schema to NoSQL Graph Data Base. In: 18th International Conference on Intelligent Systems Design and Applications (2018)
14. Sellami, A., Nabli, A., Gargouri, F.: Graph NoSQL Data Warehouse Creation. In: 22nd International Conference on Information Integration and Web-based Applications and Services (iiWAS) (2020)
15. Prat, A., Averbuch, A.: Benchmark design for navigational pattern matching benchmarking. http://ldbcouncil.org/sites/default/files/LDBC _D 3.3.34.pdf (2020)
16. Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat, A., Pham, Minh-Duc, Boncz, P.: The LDBC social network benchmark: Interactive workload. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 619–630 (2015)
17. Moalla, I., Nabli, A., Bouzguenda, L., Hammami, M.: Data warehouse design from social media for opinion analysis: the case of Facebook and Twitter. In: 13th ACS/IEEE International Conference on Computer Systems and Applications (2016)