NoSQL Graph-based OLAP Analysis

Arnaud Castelltort and Anne Laurent
LIRMM, UMR5506 - Université Montpellier 2, Montpellier, France

Keywords: OLAP Analysis, NoSQL Graph Databases.

Abstract: OLAP is a leading technology for analysing data and decision making. It helps the users to discover relevant information from large databases. Graph OLAP has been studied for several years in the OLAP framework. In existing work, the authors study how to import graph data into OLAP cube models but no work has explored yet the feasability to exploit graph structures to store analytical data. As graph databases are more and more used through NoSQL implementations (e.g., social and biological networks), in this paper we aim at providing an original model for managing cubes into NoSQL graphs. We show how cubes can be represented in graphs and how these structures can then be used for graph OLAP queries to support decision making.

1 INTRODUCTION

Graph NoSQL engines such as Neo4j are now being used more and more in applications. Their capacity to scale to large databases and complex treatments are well-suited for many applications having intensive needs for graph-oriented applications, such as in chemistry, biology, social networks, etc.

Studies have shown that these technologies present good performances, much better than classical relational databases (Board, 2013) for representing and querying such large graphs, especially for connected data.

Retrieving relevant information from such graphs in an efficient manner is a key feature. In this perspective, retrieving decisional information, as done in the OLAP framework, appears to be a promising area. OLAP allows to represent key indicators as measures (e.g., number of sales) defined over dimensions (e.g., product, time, location) and to query this information by selecting relevant pieces of information (e.g., slice and dice) or by navigating through hierarchies (e.g., from years to decades, from regions to countries), thus helping decision makers to retrieve relevant information. We thus consider adding OLAP features to NoSQL graph databases.

In the literature, works have considered coupling graph and OLAP. Models have been proposed and have been implemented. OLAP queries have been translated to the graph framework. However, as far as we know, no work has addressed the use of NoSQL graph databases for this purpose.

In this paper, we propose to study how NoSQL databases can help for retrieving relevant information from data using the OLAP paradigms. We address both the representation of data cubes through NoSQL databases and the query processing. In our work, we consider the Neo4j engine and the declarative Cypher language.

It should be noted that we do not provide any complete experimental campaign as we rather focus on the formalism of our approach in order to demonstrate that it is feasable to represent and store analytical data within graph databases.

The paper is organised as follows. Section 2 reviews the related work, namely NoSQL databases and graph OLAP databases. Section 3 introduces our proposition for modeling NoSQL graph databases while Section 4 presents the extension of Cypher queries to OLAP NoSQL data cubes.

2 RELATED WORK

We introduce below the foundation of NoSQL graph databases and the existing work on graph OLAP.

2.1 NoSQL Graph Databases

Definition 1 (Graph)**.** *A graph G is defined as a pair (V,E) where V is a set of nodes and E is a set of relations, with $E \subseteq (V \times V)$.*

Most of NoSQL graph databases are based on oriented graphs (Han et al., 2011). They contain elements (nodes and relations) together with properties on these elements. These properties can be rep-

Castelltort A. and Laurent A..
NoSQL Graph-based OLAP Analysis.
DOI: 10.5220/0005072902170224
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2014), pages 217-224
ISBN: 978-989-758-048-2

resented through $(key, value)$ pairs (Robinson et al., 2013).

Graphs are often represented graphically and are powerful for visualising and analysing data. On the one hand, they benefit from an easy human understanding. On the other hand, they offer excellent performance thanks to their data structures and query tools which are very relevant for many real-world contexts (Rodriguez and Neubauer, 2010).

Fig. 1 shows a graph and its structure in $(key, values)$ pairs.

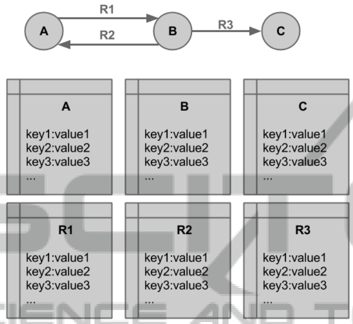


Figure 1: Properties of Nodes and Relations.

Their exist several NoSQL graph database engines (OrientDB, Neo4j,HyperGraphDB, etc.). Neo4j is recognised as being one of the top ones regarding performance (Board, 2013).

2.2 Graph OLAP

(Chen et al., 2009) has proposed to couple graphs with OLAP in 2008. This approach consists in studying how a set of several graphs can be aggregated into a single summary graph.

(Zhao et al., 2011) has introduced the concept of *Graph Cube*. This concept stands for a modelization of data cubes from graphs where dimensions are based on node attributes (e.g., age, sex, city of birth for a people) and where facts are based on countings. Fig. 2 shows the cube from the paper example considering a low level of granularity and aggregation with OLAP operators. These propositions are enhanced regarding performance in (Denis et al., 2013) with the use of distributed computer architectures.

(Li et al., 2011) proposes a method for modelizing cubes from network data, as for instance the network of co-authors. Meta-data are distinguished depending on the elements they are linked to: dimensions or facts. If they depend on dimensions, then they are considered as *informational dimension tables* as for instance spatio-temporal dimensions or as *topological dimension tables*. If they depend on facts, they are said to be *frame fact table* or *clique fact table*. Every cell in the cube contains a network. For instance, with the dimensions conference and year, the cell at position (VLDB,2010) contains the graph of all co-authors from the papers at this event. The system has been implemented in SQL Server and precomputations are possible in order to get better performance. Two types of operations are proposed: I-OLAP operations (*Informative OLAP*) when dimensions are taken from node attributes and T-OLAP (*Topological OLAP*) described in (Qu et al., 2011) when dimensions are taken from nodes and relations attributes. These operations allow to perform classical OLAP operations such as roll-up, drill-down, for instance for navigating through decades and years, slice/dice, etc.

Regarding OLAP operators, (Etcheverry and Vaisman, 2012) works on the RDF Cube vocabulary called QB to extend it. (Beheshti et al., 2012) extends the QB syntax in the so-called QB4OLAP for supporting OLAP operators directly in RDF.

(Kampgen et al., 2012) considers the execution of OLAP queries through SPARQL over RDF data. Unfortunately, the paper does not demonstrate in details how to express OLAP queries and how queries are processed.

(Petermann et al., 2014) discusses the concept of Business Intelligence in the context of graphs (storing information on commercial processes in this case, within *Business Transaction Graphs*).

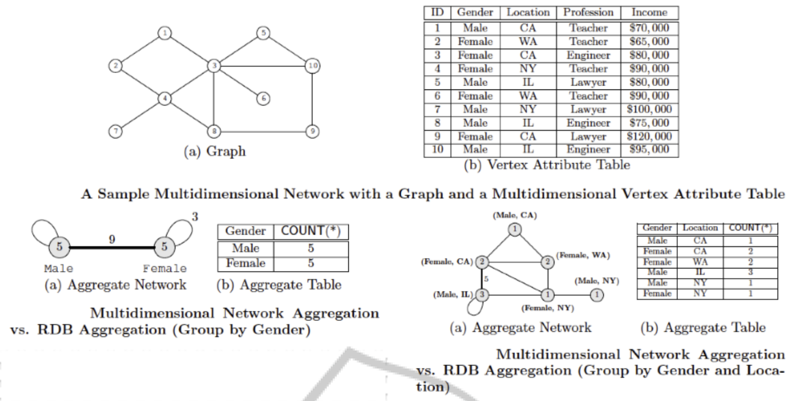
(Bachman, 2013) addresses the implementation of precomputation componants over Neo4j databases and also addresses efficient computation of node arity. This work is very important for OLAP features as many operations rely on counting incoming and outgoing relations, as for instance for counting the number of friends in a social network. However, this work is a low level implementation and does not detail how OLAP queries can be written.

Our proposition in this paper is quite different from the literature as we propose to use the graph structure as a basis for OLAP analysis.

This original contribution relies on the use of the performance and efficiency of NoSQL graph databases engines to store and query OLAP data. We first show how such NoSQL Graph OLAP data can be modelised and we then address OLAP queries over such data structures.

3 NoSQL GRAPH OLAP DATABASES

In this section, we present our contribution for modelising cubes in NoSQL graph databases. The case



(a) Graph

ID	Gender	Location	Profession	Income
1	Male	CA	Teacher	$70,000
2	Female	WA	Teacher	$65,000
3	Female	CA	Engineer	$80,000
4	Female	NY	Teacher	$90,000
5	Male	IL	Lawyer	$80,000
6	Female	WA	Teacher	$90,000
7	Male	NY	Lawyer	$100,000
8	Male	IL	Engineer	$75,000
9	Female	CA	Lawyer	$120,000
10	Male	IL	Engineer	$95,000

(b) Vertex Attribute Table

A Sample Multidimensional Network with a Graph and a Multidimensional Vertex Attribute Table

(a) Aggregate Network

Gender	COUNT(*)
Male	5
Female	5

(b) Aggregate Table

Multidimensional Network Aggregation vs. RDB Aggregation (Group by Gender)

(a) Aggregate Network

Gender	Location	COUNT(*)
Male	CA	1
Female	CA	2
Female	WA	2
Male	IL	3
Male	NY	1
Female	NY	1

(b) Aggregate Table

Multidimensional Network Aggregation vs. RDB Aggregation (Group by Gender and Location)

Figure 2: Graph Cube Model from (Zhao et al., 2011).

below serves as a running example.

Example 1. *We consider a classical database describing sales of products in cities at given dates. The dimensions are: the products described by their ProductId regrouped into Categories, the cities organised along a hierarchy City-Region-Country and the days organised along a hierarchy Day-Month-Year.*

3.1 NoSQL Graph Cubes

Our model relies on the modelisation of dimensions and facts with typed nodes. These types represent the granularity levels of the dimension, taking advantage from the best practices from NoSQL graph databases experts[1]. It should be noted that OLAP theoretical models consider such granularity levels in the literature (Gyssens and Lakshmanan, 1997). Nodes are linked by relations describing:

- links of type *hierarchy* (*HIER*) in the case of dimensions,
- links or type *fact* (*FACT*) in the case of a link from a dimension to a fact.

.

Fig. 5 shows such a representation and Fig. 4 displays an example of such a cube with three dimensions (Time, Product, City) organized through hierarchies. For instance, there have been 5 units of Sales of Canoes in New York City on March 3, 2014. Our solution presents many advantages :

- Multidimensionsal data are easily ploted on the user's screen, as demonstrated by Fig. 4. Every dimension can be colored so as to ease the user visualization.

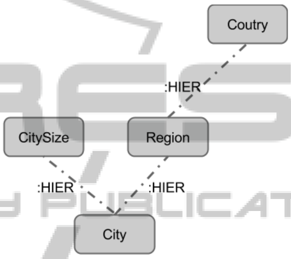


Figure 3: Modelising a dimension with multiple hierarchy in a NoSQL Graph (Neo4j).

- The nodes can be adjusted (enlarged or reduced) according to their value.
- Cells can be organized to cluster them and/or bring them closer or farther.
- Only non null cells are managed, thus avoiding sparse tables.
- OLAP navigation is eased. In particular, graph queries can be used, as well as filters that can help hiding or coloring in a different manner the data being relevant for the user (e.g., displaying only cells for low level of sales, or only sales associated to NYC). The operators are detailed below.

Complex (multiple) hierarchies are modelised in the same manner by defining several relations directed to the parent nodes, as illustrated by Fig. 3. For instance, NYC is both to the east of the US and belongs to the category of Big Cities.

Facts are represented by nodes of type :*NFACT* labeled (a label is a graph construct that is used to group nodes into sets) by the type of fact they represent and are linked to dimension nodes through relations of type :*FACT*. Only the cells for which values are known are modelised, thus copping with the problem of cube sparsity (Niemi et al., 2003).

It should be noted that the value in the node is valid only if it is considered in relation with all the di-

[1] http://blog.neo4j.org/2010/03/modeling-categories-in-graph-database.html

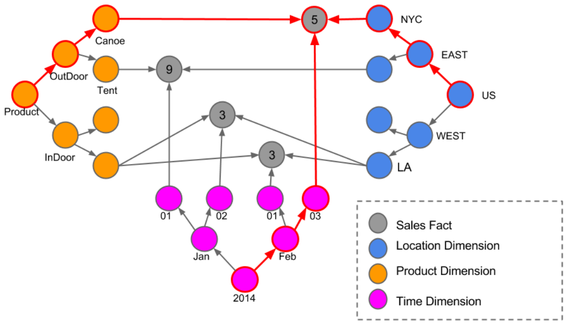


Figure 4: A NoSQL Graph OLAP Cube.

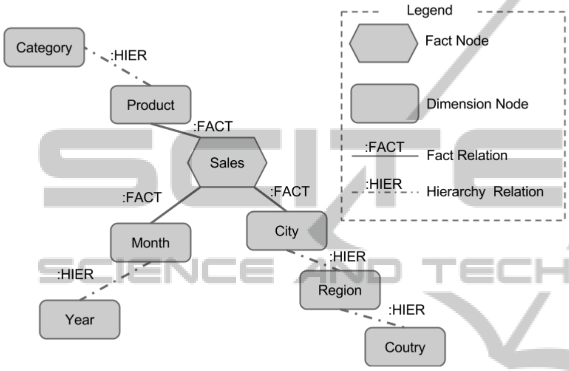


Figure 5: Modelising cubes and dimensions in a NoSQL Graph (Neo4j).

mensions. For instance, as stated above, Fig. 4 shows that there have been 5 units of Sales of Canoes in New York City on March 3, 2014. This value of 5 must be associated with all these three dimension values, i.e. *Canoes*, *NYC*, *March*3, 2014.

If a dimension is omitted, then the value must be re-computed. In such a case where one (or several) dimension is omitted, a roll up operator has been applied. In the OLAP framework, data analysis are indeed performed regarding dimensions in a so-called data cube. Sales can be analysed with respect with the three dimensions, or with respect with two dimensions out of these three, or with respect with one dimension out of the three dimensions. The combinations of dimensions are called cuboids, the set of cuboids being the cube, as shown by Fig. 6 for the Sales example. This example shows that in such a case with 3 dimensions, there are $2^3 = 8$ cuboids. All cuboids form a lattice.

More complex schemas (e.g., constellations) are easily modeled with the same concepts. We consider the following definitions for a more formal approach.

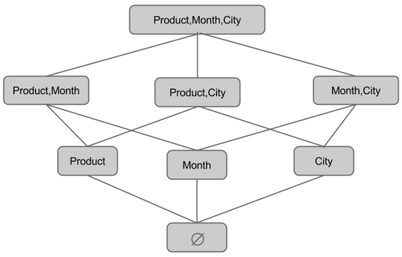


Figure 6: Lattice of Cuboids

3.2 Definitions

Definition 2 (NoSQL Graph - Dimension Level)**.** *Let G be a NoSQL graph database. A dimension level of G is defined as a possible value associated with the property key Type.*

For instance, we consider the dimension level *City*.

Definition 3 (NoSQL Graph - Dimension Node)**.** *Let G be a NoSQL graph database and $\mathcal{L}$ a set of associated dimension levels. A dimension node n is defined as a node such as $n.type \in \mathcal{L}$.*

We consider for instance the set of levels $\{City\}$ and *NYC* as a node, with $NYC.type = City$.

Definition 4 (NoSQL Graph - Dimension Relation)**.** *Let G be a NoSQL graph database and $\mathcal{N}$ a set of dimension nodes. A dimension relation r is defined as a pair $(n_1, n_2) \in \mathcal{N}^2$ such that:*

- *r.type=HIER*
- $n_1.type \neq n_2.type$.

We consider for instance:

- the set of levels $\{City, Region\}$,
- the nodes $NYC, East$,
- the types $NYC.type = City$ and $East.type = Region$, and
- the hierarchical relation $(NYC) - [: HIER] - > (East)$.

Definition 5 (NoSQL Graph - Dimension)**.** *A NoSQL graph dimension D is defined as a triplet (L, N, R) with L a set of dimension levels, N a set of dimension nodes defined over L and R a set of dimension relations defined over N.*

Definition 6 (NoSQL Graph - Type of Fact)**.** *Let G be a NoSQL graph database. A type of fact is defined as a possible value on the key TFact.*

For example, we consider the type of fact *Sales* which records the number of sales.

Definition 7 (NoSQL Graph - Fact)**.** *Let $\mathcal{F}$ be a set of types of facts. A fact is defined as a pair (n, r_d) with n a node such that $n.type = NFACT$, $n.tfact \in \mathcal{F}$ and r_d a set of dimension relations $\{r = (n,x)\ s.t.\ r.type = FACT\}_{r \in r_d}$.*

The fact is said to be defined over dimensions $\mathcal{D} = \{x.type\}$.

Definition 8 (NoSQL Graph - Cube)**.** *A NoSQL graph cube of type t is defined as*

- *the set of facts f such that $f.tfact = t$,*
- *the set of paths from the f nodes having type FACT and HIER.*

For example, the *Sales* cubes is the set of nodes where the value for key value *TFact* is *Sales* and the set of paths from these nodes traversing relations of type *FACT* and *HIER*, thus traversing (through hierarchies) nodes of type *Cities*, *Region*, *Country*, *ProductId*, *Category*, *Month*, *Year*.

3.3 Computing NoSQL Graph Cubes

OLAP engines are distinguished whether they pre-compute or not all parts of the cubes, i.e. all the cuboids from the lattice shown in Fig. 6. Models where all cuboids are pre-computed are known as MOLAP models (Multidimensional OLAP), while models where no cuboid is pre-computed are considered as ROLAP models (Relational Models). Hybrid models are possible (HOLAP) where highly requested cuboids are pre-computed while other ones are not. MOLAP engines are faster to deliver the information but have some limits as they require large volumes of memory to load pre-computations, while ROLAP engines can scale more easily as no memory is required to store pre-computations, but they are slower.

The model we propose is similar to a MOLAP representation as links are materialized. We indeed claim that the performances of NoSQL databases allow for reconsidering MOLAP models. This model has proven his capacity to scale to very large volumes of data.

As a MOLAP model, our contribution is thus very suitable for navigating through data and hierarchies.

The computation of such NoSQL graph cubes is meant to be performed by importing source data within the model. Hypergraphs may be built if the data are already in NoSQL graph databases. Due to lack of space, OLAP graph-ETL tools are not explored in this paper.

4 EXTENSION OF THE CYPHER LANGUAGE TO OLAP QUERIES

The model of OLAP cubes in NoSQL graph databases proposed above allows us to define how to navigate through such data structures in an OLAP manner. OLAP has defined several operators, the most common ones being Slice, Dice, Roll Up (Cabibbo and Torlone, 1997). Many works have addressed how to query graphs (Wood, 2012). In this paper, we consider the Cypher syntax, Cypher being the declarative query language over Neo4j.

4.1 Basic Operators

4.1.1 Slice

In OLAP, the slice operator allows to select slices from the data with respect to a condition on the dimension values. It is for instance used to reduce the data to the information related to one city (e.g., New York) or to one region (e.g., East). Note that the Cypher queries we propose below in Listings 1 and 2 return the fact with all its dimensional context.

Listing 1: Slice - Selecting Sales Results from the New York City.

```
MATCH (n)-[r1:FACT]->(x), (n)-[r2:FACT]->(y↩
    )
WHERE x.name = 'NYC'
RETURN n,r1,r2,x,y
```

Listing 2: Slice - Selecting Sales Results from the East.

```
MATCH (n)-[r1:FACT]->(x)-[h:HIER*]->(e), (n↩
    )-[r2:FACT]->(y)
WHERE e.name = 'East'
RETURN n,r1,r2,x,e,y
```

4.1.2 Dice

In OLAP, the dice operator allows to select subcubes from the data with respect to a selection condition. It is for instance used to reduce the data to the information related to sales from the cities of New York or Los Angeles and the products Canoe or Tent.

The *Dice* operator does only apply on fact nodes by specifying the possible values in the WHERE clause below in Listing 3.

Listing 3: Dice - Selecting Sales Results with a Dice Operation.

```
MATCH (n)-[r1:FACT]->(x), (n)-[r2:FACT]->(y↩
     ), (n)-[r3:FACT]->(z)
WHERE (x.name = 'NYC' or x.name = 'LA') and↩
      (y.name = 'Canoe' or y.name = 'Tent')
RETURN n,r1,r2,r3,x,y,z
```

4.1.3 Roll Up

In the case of the roll up operator, the computation of aggregated facts with respect to hierarchy levels and :*FACT* links is modelised at the upper-level. All the fact node values are aggregated by respecting the semantic and the additivity properties of the measures (Lenz and Thalheim, 2009).

For instance, the *sum* will be considered for summing up all number of sales from the fact *Sales*. A new node is created for storing the new aggregate integrate this new node at the upper level of hierarchy with relation *FACT* as shown in Listing 4.

Listing 4: Roll Up - Summing up all number of sales from the East.

```
MATCH (n)-[r1:FACT]->(x)-[h:HIER*]->(East)
WHERE East.name = 'East'
WITH sum(n.value) as aggregateValue, East
CREATE (aggregate { value: aggregateValue})↩
     -[:FACT]->(East)
RETURN aggregate, East
```

Precomputations can be considered in the same way as the cuboid materialisation in order for instance to precompute the number of sales per year, per category, per region, etc.

4.2 Implementation Issues

The model we propose can be implemented in the Neo4j graph engine.

As depicted by Fig. 7 and discussed in (Castelltort and Laurent, 2014), regarding queries, there are several possibilities to implement the extension of Cypher:

1. Creating a specific language (Cypher OLAP) extending the Cypher syntax and rewriting these queries into well-formatted Cypher queries;
2. Extending the Cypher syntax and rewriting the queries using the low level API which can interact with the Neo4j engine;
3. Extending the API with advanced querying features and proposing this extension to developers;
4. Combining the above-mentioned options for developing an OLAP extensions based on the OLAP API.

Whatever the choice may be, the type of rewriting is similar. This is the reason why we do not address this topic in this paper. However, the impacts on performances and on the user experience may be very different depending on this choice.

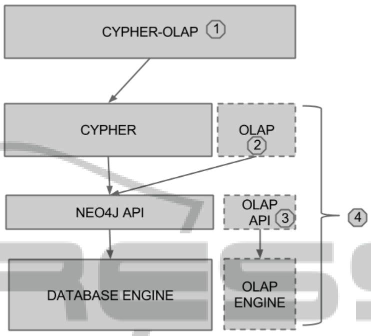


Figure 7: Alternatives for Extending the Cypher Language within Neo4j.

Cubes can be created in the Neo4j engine by respecting the model proposed in this paper. Nodes and relations are considered with their respective types, depending on their nature (facts, dimensions or hierarchy).

For every fact node, we have one outgoing chain per dimension starting with a :*FACT* relation. These Dimensions are structured within chains representing the hierarchy. For this reason, dimension nodes are linked within the chains by :*HIER* relations.

For example, Fig. 8 presents the cube from Fig. 4 that can be created in Neo4j. For instance, the gray node in the left hand side representing the 9 sales of tents in Boston at January 1st, 2014 is linked to three dimensions with :*FACT* relations.

- The first relation is linked to the *Tent* node from the *Outdoor* category and *Product* dimension.
- The second relation is linked to the *Boston* node from the *East* region of the *US* country.
- The third relation is linked to the *Jan*0114 node from the *Jan*14 month of the *Y*2014 year.

They can then be queried regarding these types and/or some keys and values for defining OLAP operations as defined above.

Fig. 9 diplays the result of the Slice operation for retrieving the sales from the East. Thus, the resulting figure contains the data reduced to the cities of Boston (9 tents sold) and New York (5 canoes sold).

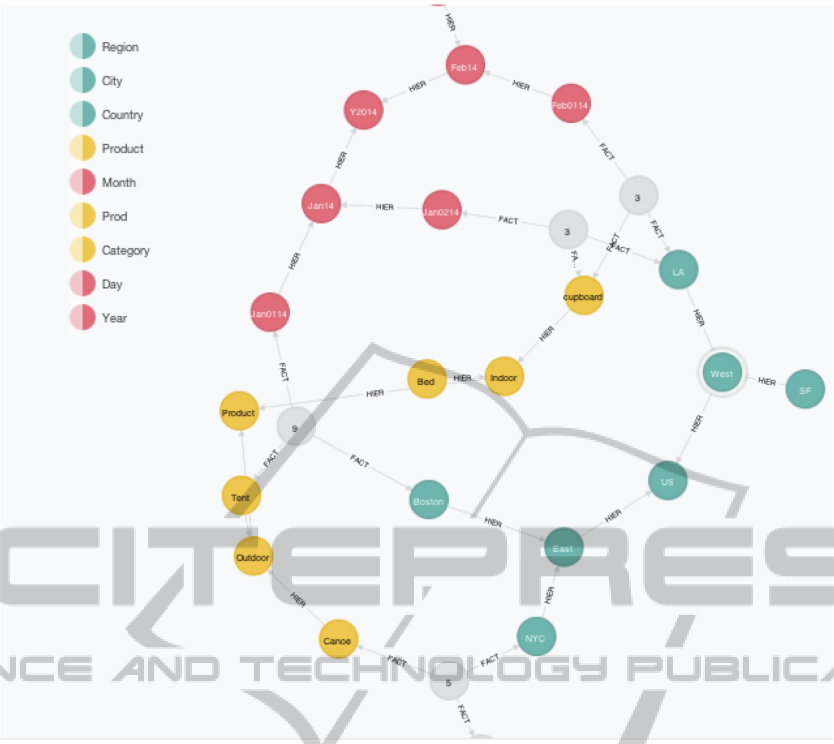
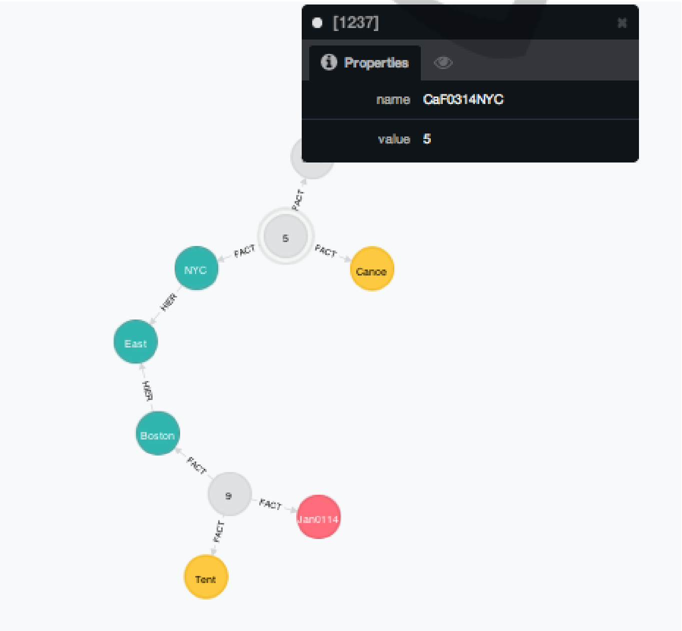


Figure 8: A NoSQL Graph Cube in Neo4J.


[1237]

Properties

name CaF0314NYC

value 5

Figure 9: A Slice Operation over the NoSQL Graph Cube in Neo4j.

5 CONCLUSION AND FURTHER WORK

In this paper, we address the topic of coupling NoSQL graph databases and OLAP. These two scientific domains are very active and important in many real-worl applications. We claim that NoSQL graph databases are a perfect candidate for supporting OLAP features, as they are both efficient and suitable for representing data in a very intuitive manner for decisional purposes. Analysing data and retrieving relevant information indeed rely on the capacity to deal with big amount of data and the capacity to assist decision making.

For this purpose, we propose an OLAP data structure based on NoSQL graph databases and we introduce OLAP queries based on the Cypher declarative language. Some tests have been performed with the Neo4j engine, showing the feasability of our proposal.

Further work will help us strengthening the implementation and tests over large databases and complex OLAP queries in order to compare existing approaches to our proposed solution. Both future processes and queries will be enhanced, by defining specific ETL componants and by implementing the extension of the Cypher language. We also aim at exploiting the NoSQL graph databases and the attributes on relationships in order to better represent and manage fuzzy hierarchies (Laurent, 2003) that are very difficult to deal with in existing engines.

REFERENCES

Bachman, M. (2013). *GraphAware: Towards Online Analytical Processing in Graph Databases*. PhD thesis, MSc Degree in Computing (Distributed Systems) of Imperial College London.

Beheshti, S.-M.-R., Benatallah, B., Nezhad, H. R. M., and Allahbakhsh, M. (2012). A framework and a language for on-line analytical processing on graphs. In Wang, X. S., Cruz, I. F., Delis, A., and Huang, G., editors, *Web Information Systems Engineering-WISE 2012*, volume 7651 of *Lecture Notes in Computer Science*, pages 213–227. Springer.

Board, T. T. A. (May 2013). Technology radar, http://thoughtworks.fileburst.com/assets/technology-radar-may-2013.pdf.

Cabibbo, L. and Torlone, R. (1997). Querying multidimensional databases. In *In Sixth Int. Workshop on Database Programming Languages*, pages 253–269.

Castelltort, A. and Laurent, A. (2014). Fuzzy queries over nosql graph databases: Perspectives for extending the cypher language. In *Int. Conf. on Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer.

Chen, C., Yan, X., Zhu, F., Han, J., and Yu, P. (2009). Graph olap: a multi-dimensional framework for graph data analysis. *Knowledge and Information System (KAIS)*.

Denis, B., Ghrab, A., and Skhiri, S. (2013). A distributed approach for graph-oriented multidimensional analysis.

Etcheverry, L. and Vaisman, A. A. (2012). Qb4olap: A vocabulary for olap cubes on the semantic web. In Sequeda, J., Harth, A., and Hartig, O., editors, *COLD*, volume 905 of *CEUR Workshop Proc.* CEUR-WS.org.

Gyssens, M. and Lakshmanan, L. V. S. (1997). A foundation for multi-dimensional databases. In Jarke, M., Carey, M. J., Dittrich, K. R., Lochovsky, F. H., Loucopoulos, P., and Jeusfeld, M. A., editors, *VLDB'97, Proc. of 23rd Int. Conf. on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 106–115. Morgan Kaufmann.

Han, J., Haihong, E., Le, G., and Du, J. (2011). Survey on nosql database. In *Proc. of the 6th Int. Conf. on Pervasive Computing and Applications (ICPCA)*, pages 363–366.

Kampgen, B., O'Rain, S., and Harth, A. (2012). Interacting with statistical linked data via olap operations. In *In Proc. of the Int. Workshop on Interacting with Linked Data*, pages 36–49.

Laurent, A. (2003). A new approach for the generation of fuzzy summaries based on fuzzy multidimensional databases. *Intell. Data Anal.*, 7(2):155–177.

Lenz, H.-J. and Thalheim, B. (2009). A formal framework of aggregation for the olap-oltp model. *J. UCS*, 15(1):273–303.

Li, C., Yu, P. S., Zhao, L., Xie, Y., and Lin, W. (2011). Infonetolaper: Integrating infonetwarehouse and infonetcube with infonetolap. *PVLDB*, 4(12):1422–1425.

Niemi, T., Nummenmaa, J., and Thanisch, P. (2003). Normalising olap cubes for controlling sparsity. *Data Knowl. Eng.*, 46(3):317–343.

Petermann, A., Junghanns, M.and Mller, R., and Rahm, E. (2014). BIIIG: enabling business intelligence with integrated instance graphs. In *5th Int. Workshop on Graph Data Management (GDM 2014)*, pages 03–31.

Qu, Q., Zhu, F., Yan, X., Han, J., Yu, P. S., and Li, H. (2011). Efficient topological olap on information networks. In Yu, J. X., Kim, M.-H., and Unland, R., editors, *DASFAA (1)*, volume 6587 of *Lecture Notes in Computer Science*, pages 389–403. Springer.

Robinson, I., Webber, J., and Eifrem, E. (2013). *Graph Databases*. O'Reilly.

Rodriguez, M. A. and Neubauer, P. (2010). The graph traversal pattern. *CoRR*, abs/1004.1001.

Wood, P. T. (2012). Query languages for graph databases. *SIGMOD Rec.*, 41(1):50–60.

Zhao, P., Li, X., Xin, D., and Han, J. (2011). Graph cube: On warehousing and olap multidimensional networks. In *Proc. of the 2011 ACM SIGMOD Int. Conf. on Management of Data*, SIGMOD '11, pages 853–864, New York, NY, USA. ACM.