A Framework for Building OLAP Cubes on Graphs

Amine Ghrab[1,2,3(✉)], Oscar Romero[3], Sabri Skhiri[1], Alejandro Vaisman[4], and Esteban Zimányi[2]

[1] EURA NOVA R&D, Mont-Saint-Guibert, Belgium
{amine.ghrab,sabri.skhiri}@euranova.eu
[2] Université Libre de Bruxelles, Brussels, Belgium
ezimanyi@ulb.ac.be
[3] Universitat Politècnica de Catalunya, Barcelona, Spain
oromero@essi.upc.edu
[4] Instituto Tecnológico de Buenos Aires, Buenos Aires, Argentina
avaisman@itba.edu.ar

Abstract. Graphs are widespread structures providing a powerful abstraction for modeling networked data. Large and complex graphs have emerged in various domains such as social networks, bioinformatics, and chemical data. However, current warehousing frameworks are not equipped to handle efficiently the multidimensional modeling and analysis of complex graph data. In this paper, we propose a novel framework for building OLAP cubes from graph data and analyzing the graph topological properties. The framework supports the extraction and design of the candidate multidimensional spaces in property graphs. Besides property graphs, a new database model tailored for multidimensional modeling and enabling the exploration of additional candidate multidimensional spaces is introduced. We present novel techniques for OLAP aggregation of the graph, and discuss the case of dimension hierarchies in graphs. Furthermore, the architecture and the implementation of our graph warehousing framework are presented and show the effectiveness of our approach.

1 Introduction

As the business and social environments become more interconnected and dynamic, graph-structured data become more prominent. Graphs have the benefit of revealing valuable insights from their topological properties. A new class of business facts and measures could be explored within the multidimensional space built from graphs. In addition, a multitude of emerging decision making problems can be represented using graph models and solved using graph algorithms. Common problems are fraud detection, trends prediction, real-time recommendation and Master Data Management just to name a few [1,2]. For example, by examining the eigenvector centrality in a social network, an analyst can detect influential people or communities. This information could then be

T. Morzy et al. (Eds.): ADBIS 2015, LNCS 9282, pp. 92–105, 2015.
DOI: 10.1007/978-3-319-23135-8_7

reused for recommendation and targeted advertising. In financial services, complex graph patterns could also be used to represent and detect complex rings which might lead to discover fraudulent transactions. Such scenarios rely mostly on the analysis of complex relationships between data entities, which is difficult to formulate and expensive to process using traditional relational systems [1]. We experience thus a growing need to integrate graph data within decision support systems. Such integration will help decision makers get an extended view and thus better understanding of their business environments and make more informed decisions.

Current decision support systems often rely on data stored in the organization's data warehouse. Data in the data warehouse is modeled following the multidimensional model, represented using the cube metaphor and interactively queried using the OLAP paradigm. However, traditional decision making systems, and particularly data warehousing solutions were initially developed to support relational data, and are not equipped for the efficient analysis and aggregation of graph properties. To extend current decision support systems with graph data and gain new insights over graphs, we need to design a novel OLAP technique aware of the specific properties of graphs.

Many approaches were proposed in the literature to extend current decision support systems with graphs [3–5]. They suggested the first foundations for building OLAP cubes on graphs. However, their techniques focused mostly on homogeneous graphs (i.e., graphs where all nodes are of the same type, and all edges are the same), and the OLAP analysis focus mainly on the graph topology as the measure of interest [3–5]. In such cases, all the attributes of the graph elements are considered as the dimensions and are used for aggregating the graph and performing its multi-perspective analysis. However, real-world graphs are complex and often heterogeneous. In this paper, we extend the state of-the-art to heterogeneous graphs (i.e., graphs where nodes and edges could be of different types to represent different real-world entities, and the different relationships between them). Therefore, not all attributes could be considered as dimensions through which the graph could be examined. We also examine a new class of measures to get additional insights from the graph topology. We extend the analysis capabilities on graphs by integrating GRAD, an analysis-oriented graph database model [6,7]. GRAD natively supports the representation of hierarchies and the analysis of the content of nodes. We use these characteristics to support dimension hierarchies and build additional OLAP cubes on graphs. We propose our novel technique for building OLAP cubes on graphs. Thereby equipping decision makers with the capability of performing effective multi-level/multi-perspectives analysis of their graph data and examining new business facts. Our main contributions in this paper are summarized as follows:

- We define the multidimensional concepts for graph data, and propose novel techniques for extracting the candidate multidimensional concepts and building graph cubes from property graphs.
- We present an extension of the property graph model, tailored for multidimensional analysis, and examine the additional candidate graph cubes brought by

this extension. We further extend our work to support dimension hierarchies within graphs.
– We suggest a graph data warehousing architecture, and provide an effective prototypical implementation of our techniques for building OLAP cubes.

The remaining of this paper is structured as follows: Sect. 2 presents our running example. In Sect. 3, we formally define the multidimensional structures on graphs. Section 4 presents our technique for extracting potential multidimensional spaces and building graph cubes on property graphs. In Sect. 5 we propose a technique for building OLAP cubes on novel graph database model, and extend our approach to support dimension hierarchies in graphs. Section 6 presents the architecture and implementation of our proof-of-concept graph warehousing framework. Section 7 discusses related work. Finally, Sect. 8 sketches future work and concludes the paper.

2 Running Example

We illustrate the analysis opportunities brought by graphs using a movie graph. The original dataset was published by the GroupLens research group.[1] The resulting graph contains movies with attributes, such as the year of release, titles, ratings and scores from different communities etc. Each movie is linked to its actors with an edge that contains the rank of the actor on the movie. We further enrich the dataset with information about actors' birth date and nationality, and movies country from the Movie Database website.[2] Figure 1(a) shows a subgraph of the movie graph. We start with a simple and flat multidimensional schema shown in Fig. 1(b). We introduce in Sect. 5 a more complete schema supporting hierarchies and enabling more advanced analysis.

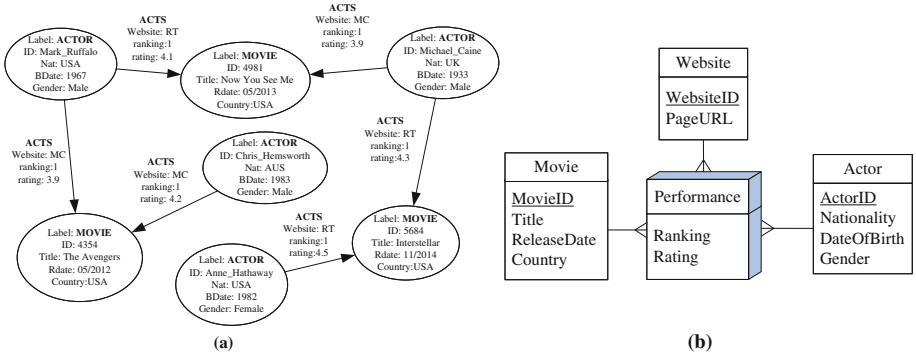

Fig. 1. A sample movie graph

[1] http://grouplens.org/datasets/movielens.
[2] https://www.themoviedb.org/.

3 Multidimensional Concepts on Graphs

In this section we formally define the multidimensional structures in the context of heterogeneous attributed graphs. We start with dimension levels.

Definition 1 [*Dimension Level*]. A level is a pair $L_i = \langle name, \mathcal{P}_i \rangle$, where *name* is the name of the level, and $\mathcal{P}$ is the aggregation pattern. $\mathcal{P} = (T, C)$ is a pair, where T is the pattern's topology and C are the constraints applied on its content (i.e. attributes). $\mathcal{P}$ is used to identify all graph elements that belong to the dimension's level and that should be merged after roll-up. □

Dimensions provide the possible perspectives for the analysis of the graph topology and content. In graphs, we distinguish two types of dimensions: (1) Node-based dimensions, which are represented by the attributes of the nodes, and (2) Edge-based dimensions, which are represented by the attribute of the edges. We define a dimension as follows:

Definition 2 [*Dimension*]. A dimension is defined as $\mathcal{D} = \langle name, \mathcal{L}, \mathcal{R} \rangle$, where $\mathcal{L} = \{L_1, ..., L_n, All\}$ is the set of the dimension levels. $\mathcal{R}$ is a partial order on the elements of $\mathcal{L}$ and describes a directed acyclic graph defining the hierarchy and the aggregation direction between the dimension's levels L_i. The base level L_1 and highest level *All* are located at the ends of the partial order. □

In the multidimensional model, a measure is the basic unit of data that is placed in the multidimensional space and examined through the dimensions.

Definition 3 [*Measures*]. A measure m is identified by the triple $\langle name, \mathcal{F}, \mathcal{A} \rangle$. It is computed over a graph $\mathcal{G} \in \mathbb{G}$ using a function $\mathcal{F}$ as follows: $\mathcal{F} : \mathbb{G} \longrightarrow Dom(m)$. In graphs, $\mathcal{F}$ could be a graph-specific function such the PageRank algorithm. $\mathcal{A}$ is the aggregation function (e.g., SUM, AVG etc.) used to compute an aggregated value of the measure. □

Multiple classification for graph measures were proposed in the literature, such as the classification by the aggregation type (i.e., distributive, algebraic and holistic) [3]. Here we propose a new classification of graph measures, based on the type and the computation algorithm.

- **Content-Based Measures:** They are extracted from the attributes of graph elements. These measures are similar to the traditional measures and do not capture the graph topology. For example, the average rating of a movie and the average rank of an actor are content-based measures.
- **Graph-Specific Measures:** They capture the topological properties of graphs and are obtained by applying graph algorithms. They could be classified according to the type of the output as either (1) *numerical*, where the output is a numerical value such as the value of the page-rank, or (2) *topological*, where the measure is represented using graph structures such as the path between a pair of nodes. The second possible classification makes the distinction between (1) *local* measures, which are computed separately for graph

nodes or edges (e.g., the centrality of an actor), and (2) *global* measures which are computed for the whole graph (e.g., the diameter or number of cycles of the graph).
- **The Graph as a Measure:** As discussed by Chen et al. in [3], the graph itself could be considered as a measure examined from different perspectives and at different aggregation levels.

The cube metaphor is widely accepted as the underlying logical construct for conventional multidimensional models. Here we define the concept of cube using the notion of aggregate graphs defined as follows.

Definition 4 [*Aggregate Graph*]. An aggregate graph $\mathcal{G}'$ of an initial graph $\mathcal{G}$ is a graph obtained by condensing a subset of the nodes and edges of $\mathcal{G}$. Hence, each node corresponds to a set of nodes in $\mathcal{G}$, and each edge is the result of fusion of edges between pairs of aggregated nodes. □

Definition 5 [*Graph Cube*]. A graph cube corresponds to a set of aggregate graphs obtained by restructuring the initial graph $\mathcal{G}$ in all possible aggregations. Each cuboid is therefore represented as an aggregate graph of $\mathcal{G}$. If an aggregation is performed from L_i to L_{i+1}, all graph elements that satisfy the aggregation pattern $\mathcal{P}_i$ are aggregated in the same node. The edges are constructed afterwards to link the pairs of nodes. Measures are then recomputed and placed on the aggregate graph. □

In the next sections, we show how these formal definitions map to the specific graph structures of each model and illustrate them with examples applied on the movie graph. We discuss how to select a valid subset of attributes as the candidate dimensions or measures, and build the different graph cubes.

4 Building OLAP Cubes on Property Graphs

Many current graph databases represent graphs using the *property graphs* model [8]. We show in this section how we can use property graphs as a first foundation for building OLAP cubes. However, since property graphs describe basic graph structures (which are simple and oriented for storage and operational workloads), their analysis capabilities are limited. For advanced multidimensional modeling and analysis, richer graph structures are needed as we show later in Sect. 5.

Property graphs describe a directed, labeled and attributed multi-graph. Formally, we define a property graph as follows:

Definition 6 [*Property Graph*]. A property graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}_v, \mathcal{L}_e, \Lambda_v, \Lambda_e)$, where:

- $\mathcal{V}$ is the set of nodes.
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges.
- $\mathcal{L}_v$ is the set of node labels and $\mathcal{L}_e$ is the set of edge labels.
- $\Lambda_v = \{a^1, a^2, ..., a^m\}$ is the set of node attributes represented as key/value pairs. Each node $v_i \in \mathcal{V}$ is associated with an attribute vector $\lambda_{v_i} = [a^1, a^2, ..., a^j]$.

– $\Lambda_e = \{b^1, b^2, ..., b^n\}$ is the set of edge attributes represented as key/value pairs. Each edge $e_i \in \mathcal{E}$ is associated with an attribute vector $\lambda_{e_i} = [b^1, b^2, ..., b^k]$. □

A node $v_i \in \mathcal{V}$ is represented as $v_i = (l_i, \lambda_{v_i})$, where $l_i \in \mathcal{L}_v$ is the label and λ_{v_i} is the set of attributes. Similarly, an edge $e_j \in \mathcal{E}$ is represented as $e_j = (v_s, v_e, l_j, \lambda_{e_j})$, where v_s and v_e are the start and end nodes respectively, $l_j \in \mathcal{L}_e$ is its label and λ_{e_i} is its set of attributes. Each node (resp. edge) on the graph has exactly one label. We introduce the concept of **class** (denoted Σ_i) to describe a set of graph nodes that share the same label. For example, in the movie graph of Fig. 1(a), we have two classes which are `MOVIE` and `ACTOR`.

Given a property graph $\mathcal{G}$ and a pair of nodes from two connected but distinct classes of nodes, we explore the candidate dimensions, measures and cubes that could be built by exploring the graph of these two classes. We denote dimensions that span across two linked classes as inter-class dimensions, defined as follows.

Definition 7 [*Inter-Class Dimensions*]. Let $\mathcal{G}$ be a property graph, and let $v_s \in \Sigma_s$ and $v_e \in \Sigma_e$ be a pair of nodes from two distinct classes. Let $e_i = (v_s, v_e, l_i, \lambda_{e_i})$ be an edge that relates v_s and v_e. The node-based dimensions are the attributes of the two nodes v_s and v_e (i.e., $\lambda_{v_s} = [a^1, ..., a^k]$ and $\lambda_{v_e} = [a^1, ..., a^l]$). The candidate edge-based dimensions are a subset of the attributes of the edge e_i (i.e., $\lambda_{e_i} = [b^1, ..., b^k]$). □

Example 1 (Analysis of Rating and Ranking of Actors per Website and Movie). Using the movie running example, the node-based dimensions are the attributes $\lambda_{v_{Movie}} = [ReleaseDate, Country]$ and $\lambda_{v_{Actor}} = [Nationality, DateOfBirth, Gender]$. For example, following the notation of Sect. 3, $\mathcal{D}_{Gender} = \langle Gender, \mathcal{L}, \mathcal{R} \rangle$, with the levels being the base level *Gender* and *ALL*. Therefore, all actor nodes could be at the base level where they all have the attribute *Gender*, and then could be grouped into two groups (i.e., a node for male actors, and a node for female actors), and finally grouped in one node regardless the gender. The edge-based dimension is represented by the $\lambda_{e_{ACTS}} = [Website]$ attribute of the *ACTS* edge relating actors and movies.

The graph lattice enumerates all possible OLAP aggregations of the graph, and is obtained by aggregating over all the inter-class dimensions. Figure 2 shows the graph lattice applied to the graph of Fig. 4, considering the dimensions of the previous example. Each node of the graph lattice represents an aggregate graph, that is, a cuboid of the graph cube. We distinguish three special kinds of aggregation on this graph (highlighted in Fig. 2), which are *Movie*-only aggregations (i.e., only movie nodes are kept not fully aggregated to the All level), *ACTS*-only aggregation and *Actor*-only aggregations.

Definition 8 [*Inter-Class Measures*]. Given a property graph $\mathcal{G}$ and a set of edges $E \subset \mathcal{E}$ relating nodes of the classes Σ_s and Σ_e, a content-based measure m_c is computed by applying an aggregation function on the attributes $([b^1, ..., b^k])$ of the edges $e_i \in E$. The graph considered as a measure is obtained following the graph lattice, and the graph-specific measures are obtained by applying a graph algorithms on $\mathcal{G}$, or one of its aggregate graphs. □

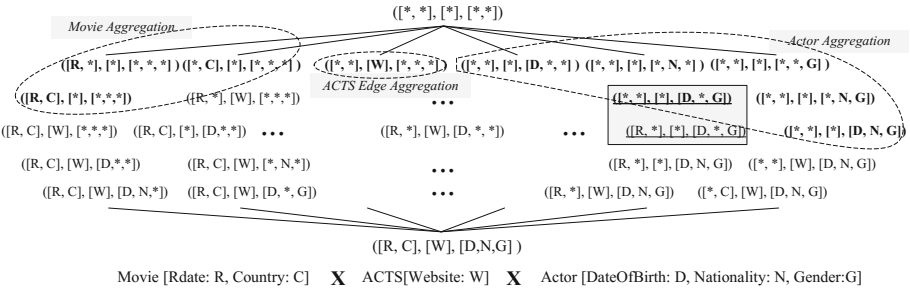

Fig. 2. The graph lattice of the movie graph

In order to analyze the properties of the relationships between the graph entities, we focus here on the potential measures existing within the edges. Clearly, we cannot assume that all attributes of the edges are dimensions. As shown by the multidimensional schema of Fig. 1(b), the attribute *Website* of the edge labeled *ACTS* could indeed be a dimension. However, the attributes *ranking* and *rating* are rather considered as measures in the current analysis scenario. We should note that the distinction between attributes that are dimensions and attributes that are measures is not straightforward, and thus requires a modeling effort from the designer to distinguish them.

Now we apply these dimensions and measures on the property graph of Fig. 1(a), and follow the graph lattice of Fig. 2 in order to study the graph cube reflecting the ranking and rating of actors in the movie graph. Figure 3(a) shows the aggregate graph (i.e., graph cuboid) where movies are grouped by

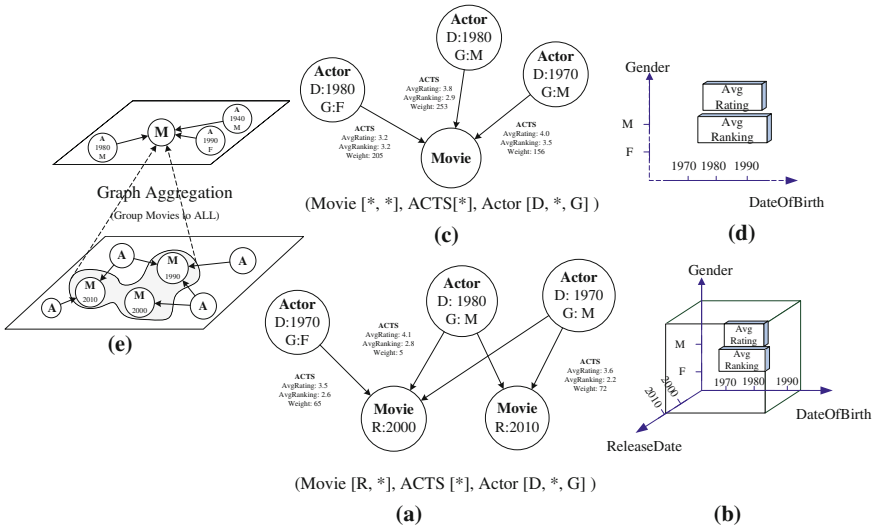

Fig. 3. OLAP aggregation of the movie graph and computation of the OLAP cubes

release date and actors are grouped by birth date and gender. A corresponding OLAP cube is shown in Fig. 3(b). The measures are *AverageRanking* and *AverageRating* of actors, which can be examined through the three dimensions left (i.e., *ReleaseDate*, *DateOfBirth* and *Gender*). We follow the graph aggregation as depicted by Fig. 3(e) to get the graph (Fig. 3(c)) and the cuboid (Fig. 3(d)) at the next aggregation level. On the lattice of Fig. 2, this aggregation corresponds to the two nodes underlined and put in rectangle. Note here that for graph-specific measures (e.g., closeness centrality of actors), the measures for the upper-level could not be computed directly from the cube at a lower level, as the computation function needs to traverse the aggregated graph itself to compute the new value of the graph-specific measure.

5 Building OLAP Cubes on GRAD

Many graph models were proposed in the literature to abstract different types of graphs and fit their particular analysis workloads [9]. In [6,7], we proposed GRAD, an analysis-oriented graph database model that extends property graphs with advanced graph structures, integrity constraints and a graph algebra. We use GRAD as the foundation for the OLAP cubes extraction techniques we present in this section.

As we discussed in the previous section, property graphs support OLAP analysis of inter-classes facts. However, they fall short from supporting OLAP analysis of the internal information stored within each node, or class of nodes. Therefore, we focus in this section on the additional cubes and analysis capabilities brought by GRAD. Note however that since GRAD extends property graphs, the candidate multidimensional spaces and cubes discussed in the previous section could similarly be built using GRAD.

5.1 OLAP Cubes on GRAD

Due to space limitations, we briefly introduce here the main components of the database model. In GRAD, we consider heterogeneous, attributed and labeled graphs. Complex attributes are supported on the nodes and rich semantics is explicitly expressed on the edges. The analysis process is centered around special analytical structures, namely *hypernodes* and *classes*. Hypernodes represent real world entities and are grouped within classes. Each analytics hypernode is an induced subgraph grouping an entity node, all its attribute and literal nodes, and all the edges between them. The core of a hypernode is the entity node which contains the label and the identifier attributes of the real world entity. Attribute nodes are attached to the entity node and denote the non-identifier, and potentially multi-valued attributes of each entity (e.g., budget, revenue). Literal nodes record the effective value of its corresponding attribute node. Rich semantics are embedded on the graph edges such as multiplicities, hierarchical and composition relationships.

Definition 9 [*GRAD Graph*]. A GRAD graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}_v, \mathcal{L}_e, \Lambda_v, \Lambda_e)$, is formally defined as follows:

- $\mathcal{V} = (V_e \cup V_a \cup V_l)$ is the set of nodes, with V_e being the set of entity nodes, V_a the set of attribute nodes, and V_l the set of literal nodes.
- $\mathcal{L}_v = \{C_i, L_a\}$ is the set of labels on entity and attribute nodes respectively.
- $\Lambda_v = \{b^1, b^2, ..., b^n\}$ is the set of entity node attributes represented as key/value pairs. Each node is associated with a vector of j attributes $[b^1, b^2, ..., b^j]$.
- $\mathcal{E} = (E_e \cup E_a \cup E_l)$ is the set of edges, with E_e being the set of entity edges, E_a the set of attribute edges, and E_l the set of literal edges. All entity edges on the graph share the same label.
- $\Lambda_e = \{b^1, b^2, ..., b^m\}$ is the set of edge attributes represented as key/value pairs. Each edge is associated with a vector of k attributes $[b^1, b^2, ..., b^k]$. □

Figure 4(a) illustrates a part of the movie graph modeled with GRAD. In this example, *Movie* is an entity node, while *Revenue* is an attribute node attached to *Movie*. The revenue has different values depending on a set of factors (location, time, language etc.), and each value is stored separately in a literal node.

In the previous section, we used property graphs to study the candidate multidimensional cubes between classes of nodes. Given a GRAD graph $\mathcal{G}$ and a class of entity nodes Σ_i, we explore in this section the candidate dimensions, measures and cubes that could be extracted from a single class Σ_i.

Definition 10 [*Intra-Class Dimension*]. Given a GRAD graph $\mathcal{G}$, a class of entity nodes Σ_i, and an entity node $u \in \Sigma_i$ with ID_u being the set of identifiers attributes of u. Then we can extract distinct sets of candidate dimensions. Each set of dimensions is the union between the attributes of the entity node and the attributes of literal edge of a given attribute node. For a given attribute node $v_i \in V_a$ linked to the entity node u, where $\lambda_i \subset \Lambda_e$ is the attributes of the literal edge $e \in E_l$ connected to v_i, $D_{v_i} = \{ID_u \cup \lambda_i\}$. □

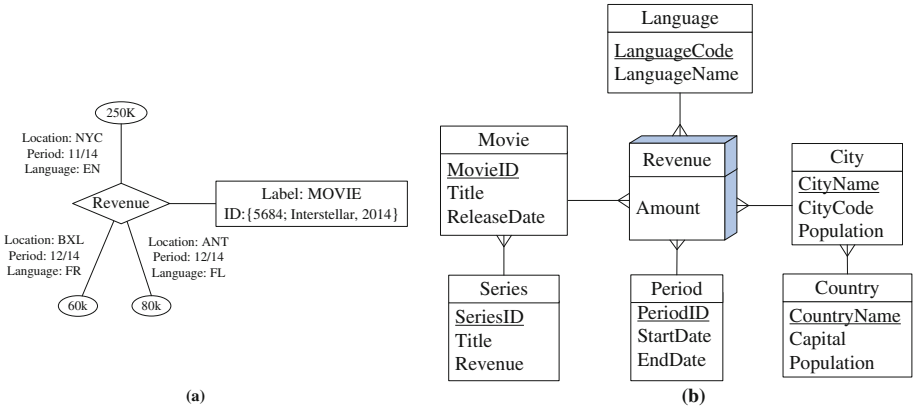

Fig. 4. Movies and actors' graph

Definition 11 [*Intra-Class Measures*]. They are defined by the triple $\langle name, \mathcal{F}, \mathcal{A} \rangle$ and are explored within each hypernode. The label of the attribute node is the name of the measure ($name \in L_a$). The actual values of these measures are embedded on the attributes of the literal nodes ($\mathcal{F}(v) \in [b^1, b^2, ..., b^k]$). □

Example 2 (Analysis of the Revenue of a Movie). Given the example of Fig. 4, suppose an analyst need to analyze the revenue of movies following the multi-dimensional schema of Fig. 4(b). Revenue is therefore considered as the name of the measure, which is the same as the label of the attribute node *Revenue*. The aggregation function is SUM. The values of the measures are stored within the literal nodes linked to the *Revenue* attribute node and the function computing the measure is the same as the one used to retrieve the value from the literal node. The dimensions for the revenue measure are named $Movie, Location, Period$, and $Language$. Given these dimensions, we can aggregate the graph to examine the value of revenue by navigating through the dimension hierarchy of the *Location* dimension from *City* to *Country* as shown in Fig. 5(a), or by rolling up to the level ALL of the language dimension as in Fig. 5(b). Concretely, at the graph level, this operation will incur merging the corresponding literal storing the measure values.

We distinguish here two types of graph aggregations: (1) Intra-hypernode aggregation, where literal nodes and edges of the same attribute node are merged, thus the dimensions is an attribute of the literal edges (e.g., revenue of a given movie by language), (2) Inter-hypernode aggregation, where entity nodes could be merged (e.g., revenue of all movies per given a city, period and language).

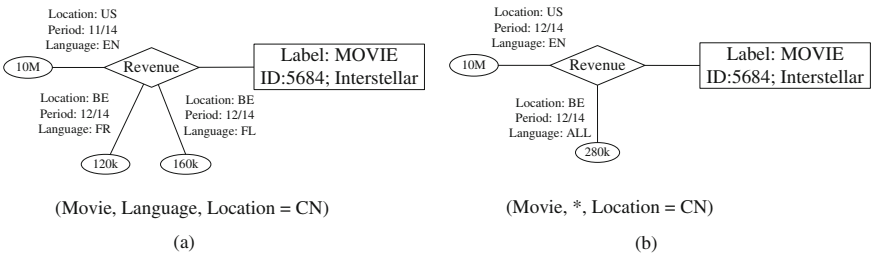

Fig. 5. Aggregation of revenue by language

5.2 Dimension Hierarchies on GRAD

In this subsection, we consider extending the OLAP analysis to support hierarchies within inter-class and intra-class dimensions.

– Dimension hierarchy for intra-class dimensions: Within each dimension (i.e., attribute location of revenue), we might have an inner hierarchy (e.g., City, Region, and Country). Therefore, we can extend the lattice with these new possible aggregations as shown in Fig. 5(a).

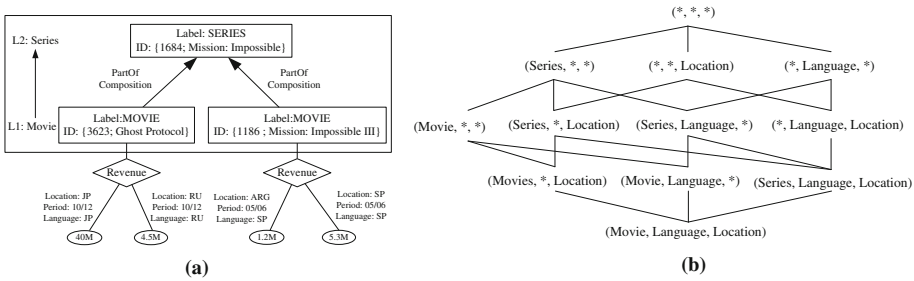

Fig. 6. Dimension hierarchy between classes

– Dimension hierarchy for inter-class dimensions: Explored between distinct classes of nodes. Within GRAD, specific types of edges such as composition and aggregation could be explicitly defined. Therefore, classes of nodes related by these specific relationships belong to the same dimension with the hierarchy following the child-parent direction of these relationships. Figure 6(a) shows the hierarchy of the movie dimension that is now composed by *Movie* and *Series* levels. The updated lattice is shown in Fig. 6(b).

6 Framework Architecture and Implementation

In this section, we present our prototypical implementation of the OLAP cubes extraction approach using Neo4j. The framework architecture is depicted in Fig. 7. The major components of our implementation are described as follows:

1. Graph ETL: The graph is extracted from external data sources that might have various formats (e.g., XML as for DBLP, or text files for MovieLens, etc.). For the running example, we have developed two modules for extracting and matching data from CSV files of MovieLens with data about actors from The Movie Database. The data is then formatted following GRAD and property graph structures before being loaded as the base graph on Neo4j.
2. Graph storage and materialization: The graph data is stored using multiple Neo4j graph database instances. We use two particular databases, one to store the graph at the base level and the other to keep the lattice. The other instances store the aggregate graphs. However, we needed a database-per-aggregate graph because Neo4j do not support materialized views on graph, and could not separate between subgraphs of the same database.
3. Graph lookup and update: This component acts as a middleware between the storage and processing layers. It loads the graph, at a given aggregation level, from a Neo4j database into HDFS to prepare it for distributed processing or aggregation. Once the processing is done, this layer stores the graph back into a new Neo4j instance if the graph was aggregated, or updates the original database if only some attributes were updated.

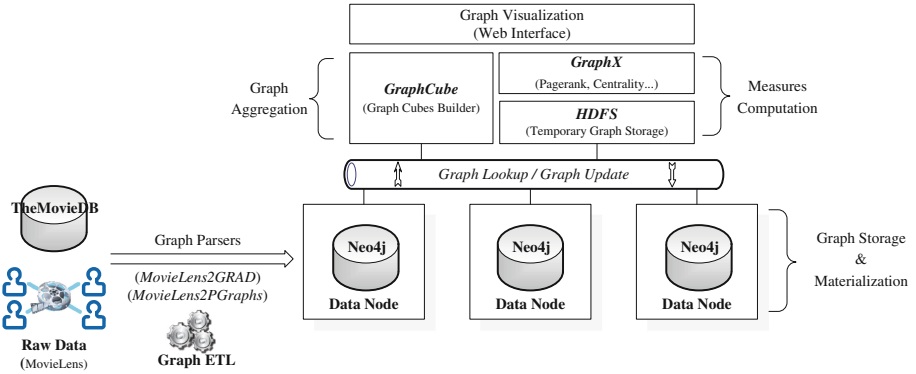

Fig. 7. Distributed OLAP cubes computation

4. Graph Aggregation and Measures Computation: Given a graph lattice, the *GraphCube* module performs the graph aggregation to generate potential graph cuboids as discussed through the paper. In order to efficiently compute of the graph-specific measures (e.g., PageRank or closeness centrality), we use the GraphX library. GraphX performs the iterative graph algorithms in-memory and thus outperforms the other distributed graph libraries on large scale graphs. Once the required graph measures are computed, the result is persisted in the corresponding Neo4j instance using the previous layer.

7 Related Work

Graph Data Warehousing: The challenge of designing graph data warehousing frameworks is part of the challenge of designing novel models and techniques for enabling multidimensional analysis of Big Data [10]. Big Data extracted from business and social environments is complex, scattered, dynamic, heterogeneous and unstructured. Most of it falls outside the decision maker's control. However, as motivated by Abelló et al. [11], incorporating such data into the decision process enables non-expert users to make well-informed decisions when required. Our work provides a foundation for extending decision support to graph data.

Graph Database Modeling: Graph database modeling and querying is the foundation for graph data warehousing. A survey of graph database models is provided by Angles et al. [9]. Multiple native graph models and query languages (e.g. GraphQL [12]) were developed to efficiently answer graph-oriented queries. In this paper, we leveraged and extended the database model we defined on [7] for graph data warehousing.

OLAP on Graphs: GraphOLAP is a conceptual framework for OLAP analysis of a collection of homogeneous graphs [3]. Attributes of the snapshots are

considered as the dimensions. Aggregations of the graph are performed by overlaying a collection of graph snapshots. Dimensions are classified as topological and informational. Informational OLAP aggregations consist in edge-centric snapshot overlaying, thus only edges changes and no changes to the nodes are made. Topological OLAP aggregations consist of merging nodes and edges by navigating through the nodes hierarchy. Qu et al. introduced a more detailed framework for topological OLAP analysis of graphs [13]. GraphCube [4] is a framework for OLAP cubes computation and analysis through the different levels of aggregations of a graph. It targets single, homogeneous, node-attributed graphs. The framework introduced the cuboid and crossboid queries for building and analyzing the different graph cubes. Distributed Graph Cube is a distributed framework for graph cubes computation and aggregation implemented using Spark and Hadoop [14]. Pagrol is a Map-Reduce framework for distributed OLAP analysis of homogeneous attributed graphs [5]. Pagrol extended the model of GraphCube by considering the attributes of the edges as dimensions. These frameworks were designed to handle homogeneous graphs [3–5]. The attributes of the graph elements are considered as the dimensions, and the graph cubes are obtained by restructuring the initial graph in all possible aggregation. Yin et al. [15] introduced a data warehousing model for heterogeneous graphs focusing on edge-based dimensions. In this paper, we extended these frameworks to the general case of heterogeneous graphs, and we discussed various techniques for building graph cubes in different settings. In [16], authors introduced a framework for OLAP on RDF data. They proposed GOLAP, a graph model for OLAP on graphs, and FSPARQL an extension to SPARQL for OLAP querying of RDF data. GOLAP introduced a rule-based approach for defining new dimensions on the graph. The same technique could be integrated, as a pre-processing phase, within our work to provide more candidate dimensions and measures.

8 Conclusion

In this paper, we proposed our contribution to graph warehousing by designing novel techniques for building OLAP cubes on graphs. We applied our approach on both property graphs and a more advanced graph database model tailored for multidimensional modeling. We discussed techniques for OLAP aggregation of the graph and tackled the case of dimension hierarchies in graphs. In addition, we provided an overview of the architecture and implementation of our graph warehousing framework.

Graph data warehousing is an emerging research field that brings various challenges similar to traditional data warehousing (e.g. high dimensionality and cubes materialization). However, the structural properties and unstructured nature of graphs calls for the development of novel modeling and processing paradigms. Our immediate future work is to enable multidimensional concepts discovery on graphs within our framework. Yet, many remaining research directions are worth investigating to build industry-grade graph warehousing systems. Among these directions we cite OLAP analysis of dynamic graphs and the definition of a proper OLAP algebra and query language for graphs.

References

1. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media Inc, Sebastopol (2013)
2. Petermann, A., Junghanns, M., Müller, R., Rahm, E.: Graph-based data integration and business intelligence with biiig. Proc. VLDB Endow. **7**(13), 1577–1580 (2014)
3. Chen, C., Yan, X., Zhu, F., Han, J., Yu, P.S.: Graph OLAP: a multi-dimensional framework for graph data analysis. Knowl. Inf. Syst. **21**(1), 41–63 (2009)
4. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 853–864. ACM (2011)
5. Wang, Z., Fan, Q., Wang, H., Tan, K.L., Agrawal, D., El Abbadi, A.: Pagrol: parallel graph olap over large-scale attributed graphs. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 496–507, March 2014
6. Ghrab, A., Skhiri, S., Jouili, S., Zimányi, E.: An analytics-aware conceptual model for evolving graphs. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 1–12. Springer, Heidelberg (2013)
7. Ghrab, A., Romero, O., Skhiri, S., Zimányi, E.: Analytics-Aware Graph Database Modeling, Technical report (2014) . http://research.euranova.eu/scientific-publications
8. Rodriguez, M.A., Neubauer, P.: Constructions from dots and lines. Bull. Am. Soc. Inf. Sci. Technol. **36**(6), 35–41 (2010)
9. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Comput. Surv. **40**(1), 1:1–1:39 (2008)
10. Cuzzocrea, A., Bellatreche, L., Song, I.Y.: Data warehousing and OLAP over big data: current challenges and future research directions. In: DOLAP 2013 Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP, pp. 67–70. ACM, New York (2013)
11. Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J.N., Naumann, F., Pedersen, T.B., Rizzi, S., Trujillo, J., Vassiliadis, P., Vossen, G.: Fusion cubes: towards self-service business intelligence. IJDWM **9**(2), 66–88 (2013)
12. He, H., Singh, A.: Query language and access methods for graph databases. In: Aggarwal, C.C., Wang, H. (eds.) Managing and Mining Graph Data. ADS, vol. 40, pp. 125–160. Springer, Heidelberg (2010)
13. Qu, Q., Zhu, F., Yan, X., Han, J., Yu, P.S., Li, H.: Efficient topological OLAP on information networks. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 389–403. Springer, Heidelberg (2011)
14. Denis, B., Ghrab, A., Skhiri, S.: A distributed approach for graph-oriented multidimensional analysis. In: IEEE International Conference on Big Data, pp. 9–16 (2013)
15. Yin, M., Wu, B., Zeng, Z.: HMGraph OLAP: a novel framework for multi-dimensional heterogeneous network analysis. In: Proceedings of the 15th International Workshop on Data Warehousing and OLAP, pp. 137–144. ACM (2012)
16. Beheshti, S.-M.-R., Benatallah, B., Motahari-Nezhad, H.R., Allahbakhsh, M.: A framework and a language for on-line analytical processing on graphs. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 213–227. Springer, Heidelberg (2012)