Noname manuscript No.
(will be inserted by the editor)

TopoGraph: An End-To-End Framework to Build and Analyze Graph Cubes

Amine Ghrab · Oscar Romero · Sabri Skhiri · Esteban Zimányi

Received: date / Accepted: date

Abstract Graphs are a fundamental structure that provides an intuitive abstraction for modeling and analyzing complex and highly interconnected data. Given the potential complexity of such data, some approaches proposed extending decision-support systems with multidimensional analysis capabilities over graphs. In this paper, we introduce TopoGraph, an end-to-end framwork for building and analyzing graph cubes. TopoGraph extends the existing graph cube models by defining new types of dimensions and measures and organizing them within a multidimensional space that guarantees multidimensional integrity constraints. This results in defining three new types of graph cubes: property graph cubes, topological graph cubes, and graph-structured cubes. Afterwards, we define the algebraic OLAP operations for such novel cubes. We implement and experimentally validate TopoGraph with different types of real-world datasets.

Keywords Graph Cube; OLAP Cube; Graph Processing; Graph Mining; Multidimensional Graph

A. Ghrab
EURA NOVA, Mont-Saint-Guibert, Belgium
E-mail: amine.ghrab@euranova.eu

O. Romero
Universitat Politècnica de Catalunya, Spain
E-mail: oromero@essi.upc.edu

S. Skhiri
EURA NOVA, Mont-Saint-Guibert, Belgium
E-mail: sabri.skhiri@euranova.eu

E. Zimányi
Université Libre de Bruxelles, Belgium
E-mail: ezimanyi@ulb.ac.be

1 Introduction

In the growing data-driven market, organizations are thriving to improve their decision-making using various data analytics frameworks. Graph analytics in particular are emerging as a promising asset for the modeling and analysis of highly complex network data generated by increasingly interconnected business and social environments. Indeed, graphs are a fundamental structure that provides an intuitive abstraction for modeling multiple complex domains such as social/professional networks, transportation networks, and protein-protein interaction networks. Graph algorithms have been used to address complex business applications such as detecting frauds in financial transactions or improving recommendation accuracy in retail (Russell, 2013; Akoglu, Tong, & Koutra, 2015). To gain novel insights into complex graph structures, this paper investigates the usage of multidimensional models and OLAP cubes to enable complex ad-hoc querying of aggregated and consolidated graph data.

Currently, the majority of the existing multidimensional models and systems were designed to support relational data. Due to the fundamental difference between graph and relational data, these models and systems are not suitable for the efficient management and analysis of graphs. Analyzing a graph from a relational perspective incurs in a potential loss of the graph structure and leads to limited analysis capabilities. Therefore, graph warehousing is emerging as the field that extends current information systems with large graph management and analytics capabilities. To effectively explore graph data warehouses, graph cubes are the building-block that enables the synthesis and complex interactive querying of large volumes of graph data. Yet, there is still room for improving the current state of the art for designing and building graph cubes, since most graph warehousing solutions ignore the topological aspects of the graphs.

Many approaches were proposed to address the graph data warehousing challenge (Ghrab, Romero, Jouili, & Skhiri, 2018; Queiroz-Sousa & Salgado, 2019). These efforts laid the foundation for multidimensional modeling and analysis of graphs. They introduced the notion of graph cube, where a subset of the graph attributes is considered as dimensions along which the graph is aggregated, and the aggregate graph itself is the measure. They studied new materialization techniques, and formalized new graph operations such as cross-boids (Zhao, Li, Xin, & Han, 2011). In this work, we extend the state of the art on graph warehousing by introducing new types of graph cubes that leverage both the content and the topology of the graphs, and expose topological and graph-structured insights. Most state-of-the-art papers consider the aggregate graph as the only measure to be examined. A notable difference of our work with regard to these paper is that we (1) capture new types of measures at a finer granularity level, (2) represent them individually with numerical values or graphs, and (3) position them within new types of graph cubes. Therefore, these cubes embed new types of measures and dimensions not captured by previous work. We discuss the required multidimensional integrity constraints on graphs, completely overlooked by current approaches, and show that our

cubes guarantee them. To the best of our knowledge, our framework is the first to define and guarantee the multidimensional integrity constraints on graphs. We further discuss the correspondence between graph cubes and relational OLAP cubes and identify the few specific cases where a graph cube could be loaded into a ROLAP cube. We show that the integration of graph cubes with the existing ROLAP systems is not a trivial task and motivate the need for graph-specific warehousing systems.

The research questions we address in this paper are: *given a graph, what kind of new graph cubes could be extracted from it? When could a graph cube be mapped and loaded to a ROLAP cube? And how could such novel graph cubes be analyzed from a multidimensional perspective?* As a result, we present **TopoGraph**, a graph data warehousing framework that extends current graph warehousing models with new types of cubes and queries combining graph-oriented and OLAP querying. TopoGraph goes beyond traditional OLAP cubes, which process value-based grouping of tables, by considering in addition the topological properties of the graph elements. And it goes beyond current graph warehousing models by proposing new types of graph cubes. These cubes embed a rich repertoire of measures that could be represented with numerical values, with entire graphs, or as a combination of them. Moreover, we discuss the correspondence between the graph cubes proposed in this paper and traditional OLAP cubes, and motivate the need for native graph warehousing systems. Given the proposed cubes, TopoGraph aims at providing answers to complex questions, asked in a business context, that require the analysis of both the content and the topology of the graph. Relevantly, TopoGraph is our proposal to overcome the current shortcomings of graph warehousing approaches resulting from our experience in real-life enterprise settings. We illustrate in the following example three typical questions to which TopoGraph is designed to answer.

Example 1 (Social Network) Consider a social network such as Twitter where a set of users are following each other, post and retweet a set of tweets as illustrated in Figure 1. We distinguish three types of queries that could be used for analyzing graph properties from a multidimensional perspective, and illustrate them on a social network through the following examples.

Content Query Content query target content-based measures and are computed by applying aggregation functions such as count and average. They are used to answering queries such as (1) counting the total number of favorites a tweet received from a certain user location, (2) the average number of followers a community of users have, or (3) the total number of tweets on a given date by users in a certain language. Existing OLAP frameworks are designed to handle this type of queries.

Topological Query These queries focus on the topological properties of graph elements, and are computed by applying graph algorithms that output numerical values, such as node degree, graph diameter, and PageRank.

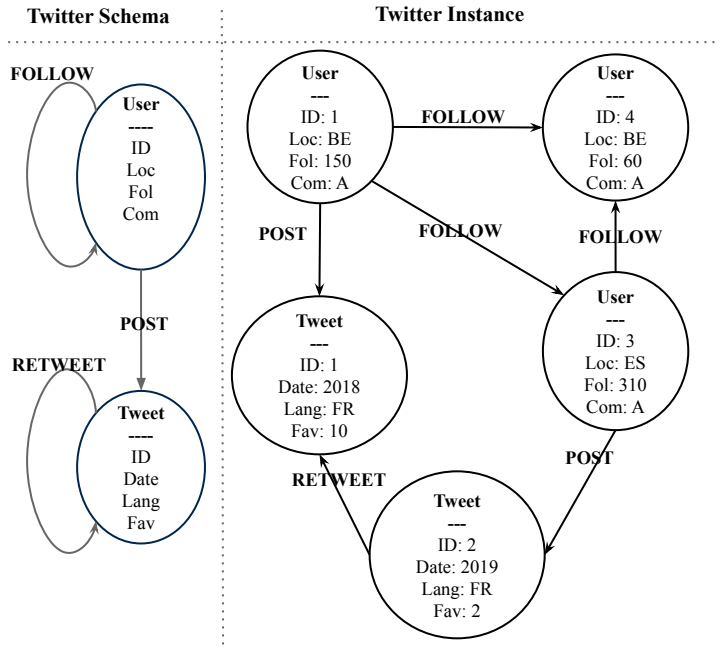

Fig. 1: Schema and Instance of a Social Network Property Graph

These queries take as input a graph and return a numerical value. They could be used to answer questions such as finding the most influential (i.e., having highest PageRank) users from a given community (i.e., after computing a community detection algorithm), who tweeted in a certain language. This kind of queries is specific for graph cubes, and they cannot be naturally supported by traditional OLAP frameworks. Graph analytics frameworks are the natural choice to tackle this type of queries, but current graph warehousing methods do not provide foundations on how to combine them with OLAP.

Graph-Structured Query Graph structured queries use graph patterns to match and retrieve complex graph information. Both the input and the output of these queries are graphs. These queries are computed by applying graph algorithms that output graphs, such as frequent pattern mining and minimum spanning tree to capture complex grouping of graph elements. They could be used to answer questions such as (1) finding the most frequent communication pattern in a network of users from a given location, or (2) retrieving the shortest path between a pair of tweets during a certain period. Similar to the previous case, current graph warehousing approaches do not support this kind of queries. □

Our main contributions are summarized as follows:

- We propose TopoGraph, a novel graph warehousing model aware of the topological properties of graphs, to support decision-making on graphs. Thus, we formally define three novel categories of graph cubes: property graph cubes, topological graph cubes, and graph structured cubes. These cubes capture both content and topological properties of heterogeneous graphs. They define new types of measures and dimensions specific for graphs and place them within the multidimensional space while preserving the multidimensional integrity constraints on graphs.
- We discuss the few cases where the information captured by the proposed graph cubes could be loaded into a corresponding OLAP cubes. We show that there is a big gap between graph cubes and relational OLAP cubes, and motivate the need for dedicated graph warehousing systems.
- We define the algebraic OLAP operations for the new graph cubes, and illustrate their application for querying the topological information embedded in the graphs. These operators enable complex graph querying and OLAP analysis of the topology and content of graph cubes from different perspectives and through different aggregation levels.
- We implement the novel graph cube computation and aggregation approach proposed by TopoGraph, and experimentally validate its efficiency with different types of real-world datasets. We further describe the architecture and the API of a social network analysis framework built with TopoGraph.

We organize the rest of the paper as follows. In Section 2 we formally define the property graph cube and its related multidimensional concepts. In Section 3, we define the concept of topological graph cubes and detail the particular process of deriving OLAP Cubes containing topological measures. Section 4 introduces graph structured cubes. We define OLAP operations on graph cubes in Section 5. In Section 6, we describe the architecture of a graph warehousing system using TopoGraph and evaluate its performance on Neo4j using multiple datasets. Section 7 discusses the related work. Finally, Section 8 discusses open challenges and concludes the paper.

2 Graph Cubes on Property Graphs

In this section, we discuss the graph data model, and we introduce the multidimensional model for graphs, used for building and analyzing the graph cubes.

2.1 Property Graphs

We use the *property graph* model to represent the input graph data. Property graphs are widely used by industrial graph databases. The model describes a directed, labeled and attributed multi-graph (Rodriguez & Neubauer, 2010). Each real-world entity is represented by a node. Relationships between entities are represented using edges. We formally define a property graph as follows:

Definition 1 [**Property Graph**] A property graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$, where:

- $\mathcal{V}$ is a finite set of nodes.
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a finite set of edges.
- $\mathcal{L} = \{l_1, ..., l_i\}$ is the set of labels of the graph elements. The label denotes the "type" of the node/edge (i.e., the class to which it belongs). $\mathbb{L} : (\mathcal{V} \cup \mathcal{E}) \rightarrow \mathcal{L}$ is a total function that returns the label of the graph element. For example, if $v \in \mathcal{V}$, then the label l of v is given by $l = \mathbb{L}(v)$.
- $\mathcal{A} = \{A_1, .., A_j\}$ is the set of attributes of the graph elements. For each node $v \in \mathcal{V}$ (resp. edge $e \in \mathcal{E}$) the set of its k attributes is $\mathcal{A}(v) = \{A_1(v), A_2(v), ..., A_k(v)\}$. □

Figure 1 shows an example of a social network property graph. The graph represents a set of Twitter user nodes and their tweets. In addition to its attributes (e.g., Location or Language), each graph element has a label, which is the distinctive attribute that defines the type of the graph element (e.g., User, Tweet, FOLLOW etc.). To be uniquely identified, each node has an identifier attribute, while edges are identified by the triple $\langle source, label, target \rangle$.

2.2 Property Graph Cubes

Using the property graphs and their aggregations, we introduce in this section the definitions of the multidimensional concepts in the context of graph data. These definitions are based on and extend the graph cube definitions presented in the previous works (Zhao et al., 2011). We formalize the concept of dimension, measure, and graph cube. The multidimensional concepts introduced in the section focus on capturing the content-based information of the graph and are therefore similar to the traditional multidimensional concepts used by relational frameworks.

Definition 2 [**Dimension**] A dimension provides the possible perspectives for the multidimensional analysis of the graph topology and content. It is defined as $D_i \in \mathcal{D}$, where $\mathcal{D} \subseteq \mathcal{A}$ is a subset of the attributes of the graph elements. Dimension attributes have to belong to a discrete domain. Given a graph element $u \in \mathcal{V} \cup \mathcal{E}$, the set of its n dimensions is $\mathcal{D}(u) = \{D_1(u), ..., D_n(u)\}$. □

Definition 3 [**Dimension Hierarchy**] A set of dimensional attributes might be linked by a hierarchy defined by the triple $\langle name, \mathcal{H}_{dim}, \mathcal{R} \rangle$. $\mathcal{H}_{dim} = \{D_i, ..., D_n, Apex\}$ represents the set of the hierarchical dimensional levels of a dimension dim. $\mathcal{R}$ is a partial order on the elements of $\mathcal{H}_{dim}$ and describes a directed acyclic graph defining the hierarchy between the dimension's levels. The base level and highest level $Apex$ (denoted with $*$) are located at the end of the partial order. □

Definition 4 [**Measure**] A measure is the basic unit of data that is placed in the multidimensional space and is examined through the dimensions. It is

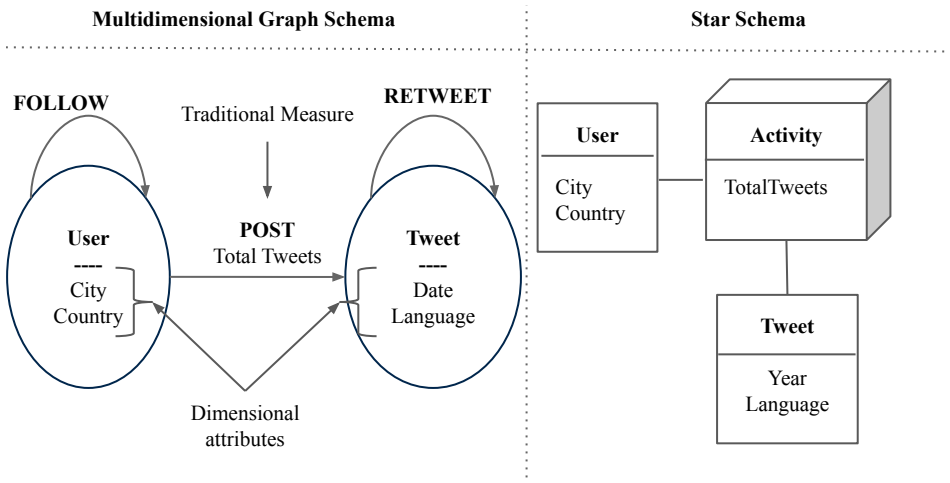

Fig. 2: Tweeting Activity Multidimensional Graph and Star Schema

defined as $M_i \in \mathcal{M}$, where $\mathcal{M} \subseteq \mathcal{A}$, and each M_i is the result of applying an aggregation function $\mathcal{F}$ (e.g., SUM, AVG etc.) on a set of attributes of the graph. Given a graph element $u \in \mathcal{V} \cup \mathcal{E}$, the set of its n measures is $\mathcal{M}(u) = \{M_1(u), ..., M_n(u)\}$. □

Multiple classification for measures were proposed in the literature (Vaisman & Zimányi, 2014), such as the classification by the aggregation function. Depending on the aggregation function a measure could be (1) distributive (i.e., the function could executed in a distributive way such as count or sum), (2) algebraic (i.e., the function is a combination of distributive ones such as average), or (3) holistic (the measure needs to be recomputed from scratch such as the median).

Definition 5 [**Multidimensional Graph**]
A multidimensional graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{D}, \mathcal{M})$ is a property graph annotated with multidimensional concepts. That is, dimensional semantics are added to the aggregated graph elements by selecting the attributes that are considered as dimensions, and those considered as measures such that $\mathcal{D} \cup \mathcal{M} = \mathcal{A}$ and the set of dimension and measure attributes are disjoint $\mathcal{D} \cap \mathcal{M} = \emptyset$. □

An aggregate graph $\mathcal{G}'$ of a multidimensional graph $\mathcal{G}$ is obtained by merging a subset of the nodes and/or the edges of $\mathcal{G}$. Only nodes (resp. edges) with the same labels can be merged together. The measures of the aggregate nodes (resp. edges) are computed using an aggregation function applied on the corresponding attributes of the nodes (resp. edges) of the initial graph. Formally, multidimensional graph aggregation is defined as follows:

Definition 6 [**Multidimensional Aggregate Graph**] A multidimensional aggregate graph is obtained by aggregating a multidimensional graph along

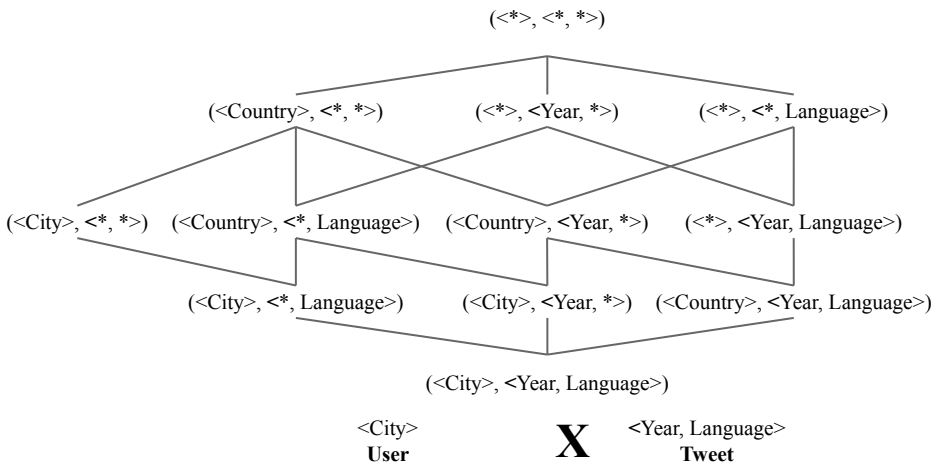

Fig. 3: Tweeting Activity Lattice

a subset of its dimensions $\mathcal{D}'$. The aggregate multidimensional graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{L}', \mathcal{D}', \mathcal{M}')$ is defined as follows:

- $\mathcal{V}'$ is the set of nodes, where each node $v' \in \mathcal{V}'$ is an aggregate node associated with a group of vertices $G_v \subseteq \mathcal{V}$ such that $\forall v \in G_v = \{v_i, .., v_k\}$ there exists one and only one v' corresponding to the set of nodes G_v.
- $\mathcal{E}' \subseteq \mathcal{V}' \times \mathcal{V}'$ is the set of edges, where each edge $e' = (u', v') \in \mathcal{E}'$ is an aggregate edge associated with a group of edges $G_e = \{e_i, .., e_k\}$ such that $\forall e(u, v) \in G_e$, there exists one and only one e' corresponding to G_e.
- The dimensions of the aggregate graph are $\mathcal{D}' \subset \mathcal{D}$. $D'_i = D_i$ if the attribute D_i was not aggregated (e.g. community remains the same after grouping users by community), and $D'_i = \emptyset$, often represented with a $*$ in the literature, if the dimension D_i is removed after the aggregation (e.g., user ages are removed after grouping users by community)
- The measures of the aggregate graph are $\mathcal{M}'$. $\forall M'_i \in \mathcal{M}' = \{M_1{}', M_2{}', ..., M_n{}'\}$, the value of the attribute M'_i of an aggregate node (resp. edge) v' is computed by applying an aggregation function $\mathcal{F}$ on the corresponding attribute values of all the nodes of G_v (corresponding to v') with $M'_i(v') = \mathcal{F}_i(G_v)$ (e.g., computing the total number of followers per community node after grouping users by community). □

Definition 7 [**Property Graph Cube**] A property graph cube is the fundamental structure supporting the multidimensional modeling and analysis of the graph data. It consists of multiple graph cuboids, each of which is a multidimensional aggregate graph built by aggregating the original multidimensional property graph along the dimensional attributes. The lattice is used to represent and organize all the possible multidimensional aggregations of the graph. Graphs cuboids relate between them when a cuboid contains

an attribute with a roll-up relationship, i.e., belong to the same dimension hierarchy and are directly related. Given n dimensional attributes, the graph cube contains 2^n graph cuboids that could be aggregated following the lattice structure. We distinguish two particular graph cuboids: (1) the base graph cuboid (where the multidimensional graph is at the base level), and (2) the apex graph cuboid (where the multidimensional graph is aggregated to the top level). □

Multidimensional aggregation of a property graph is the operation of consolidating a set of graph elements into a single one located at a higher level of the lattice. Two constraints need to be enforced when building graph cubes: (1) correct aggregation of the graph cuboids, and (2) correct placement of the graph measures within the multidimensional space. To ensure a correct aggregation of cube measures along dimension hierarchies, the graph aggregation should satisfy three constraints (Lenz & Shoshani, 1997):

- Completeness: Dimensional concepts are embedded within the graph. Therefore, every graph element should be involved in at least one dimension hierarchy. During a multidimensional aggregation, all matched graph elements should be aggregatable to a higher dimension hierarchy level. This constraint is satisfied if every graph element is associated to at least one dimension level.
- Disjointness: Each graph element is included at most once to create an aggregate entity. This condition is satisfied if every graph element could not belong to more than one dimension level at once.
- Compatibility: Compatibility between the aggregation algorithm and the aggregate graph elements to prevent non-meaningful operations such as computing the sum of user ages. The compatibility depends on the application. For example, when aggregating a group of users, the designer can decide whether applying an aggregation function such as average of the PageRank is meaningful for the application. Typically, compatibility requires additional external knowledge to know what metrics can be aggregated with what functions.

Completeness and disjointness are guaranteed by a one-to-one relationship between aggregatable and aggregate elements between consecutive cuboids. TopoGraph cubes guarantee this constraint as there exist *one and only one* node $v' \in \mathcal{V}'$ (resp. edge) in an aggregate graph corresponding to any given set of nodes G_v in the original graph. Compatibility cannot be automatically checked unless additional information is provided. On the other hand, the placement constraint is guaranteed given that the set of dimensional values generating the multidimensional space is different for each graph measure. Therefore, at most one graph element that contain a certain combination of dimension values exist in a given graph cuboid. This constraint is enforced by the statement that any pair of graph elements of the same class that have the same A'_i are merged together when aggregating a graph. In the same vein, the following types of graph cubes introduced in this paper satisfy the

multidimensional integrity constraint, as they follow a similar methodology for building and aggregating the graph.

Example 2 (Popularity Graph Cubes) Given the Twitter property graph of Figure 1, we design a possible multidimensional graph schema reflecting the tweeting activity of users. Figure 2 depicts the multidimensional graph schema and the corresponding star schema of the OLAP cube, while Figure 3 depicts its lattice. The dimensions for *User* and *Tweet* nodes are $\mathcal{D} = \{City, Country, Year, Language\}$, with the hierarchical levels $\mathcal{H}_{location}$ = {City, Country}. The measure is computed on the *POST* edge $\mathcal{M} = \{TotalTweets\}$. □

3 Topological Graph Cubes

The analysis of content-based properties of graph data (e.g., compute the average number of favorites of tweets of a given user group) is similar to the OLAP analysis of relational data in that it does not exploit the graph structure. We focus therefore in the following sections on the two graph-specific cubes introduced in Section 1: topological and graph-structured cubes.

3.1 Topological Cube Model

A rich repertoire of algorithms was developed to efficiently solve questions such centrality of nodes, or community of users. These techniques can reveal interesting properties about the graph topology and the connectivity between graph elements. Indeed, modeling data as a graph is typically done when there is an interest in exploiting such techniques. We define in this section the concept of topological graph cubes, and use them to model and analyze topological graph properties from a multidimensional perspective. As a consequence, this kind of cubes merge graph analytics and OLAP. We define first the topological concepts and discuss how to derive them from a given property graph.

Definition 8 [**Topological Attribute**] Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$, a topological attributes is defined as ${A_i}^t \in \mathcal{A}$. The value of the topological attribute ${A_i}^t$ for a node $v \in \mathcal{V}$ (resp. an edge $e \in \mathcal{E}$) is given by ${A_i}^t(u) = \mathcal{T}(v, l)$ where:

- $\mathcal{T}$: is the function computing the topological attribute value for v (resp. e). This function relies on a graph algorithm (such as Louvain for community detection)to compute the value of the topological attribute ${A_i}^t$ for the node v.
- $l \in \mathcal{L}$: most graph algorithms are designed to traverse a homogeneous graph to compute the topological attributes. However, this paper addresses the general case of heterogeneous graphs. The label l is used to guide the algorithms through a homogeneous subgraph of the input graph. □

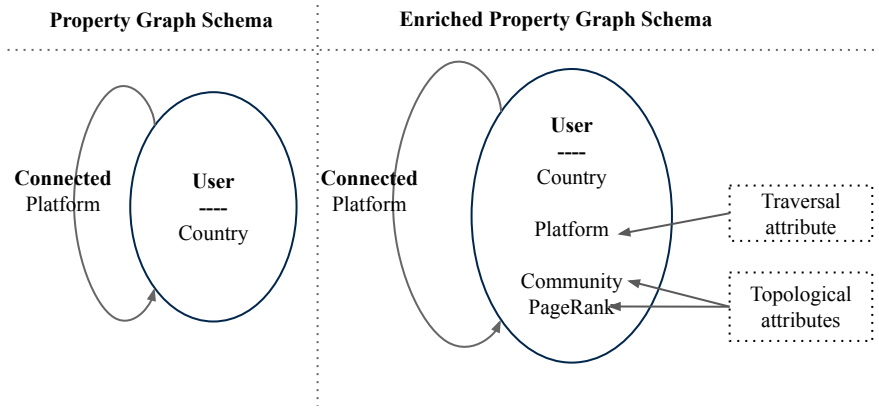

Fig. 4: Enriching Property Graph with Topological Attributes and Deriving Multidimensional Schema

Given a property graph, an enriched graph could be obtained by applying graph algorithms to add more topological properties to the nodes and edges prior performing OLAP. For example, Figure 4 shows the schema of the initial property graph, and an enriched version of it where two topological attributes were computed and integrated into the graph. We distinguish three different categories of attributes:

- Traditional attributes: reflect content-based properties of the graph elements such age *Country* of users.
- Topological attributes: reflect topological properties of the graph elements such as *community* and *PageRank* of users.
- Traversal attributes: they contain the label of the edge traversed by the graph algorithm to compute the topological attributes (e.g. *Platform* for user nodes used for computing community and PageRank).

Definition 9 [**Topological Dimensions**] Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$, the topological dimensions $D^t \in \mathcal{D}$ are a subset of the topological attributes $D^t \subseteq \mathcal{A}^t$ used for analyzing the topological graph properties from different perspectives and at different granularities. □

Definition 10 [**Topological Measures**] Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$, the topological measures $M^t \subseteq \mathcal{M}$ are a subset of the topological attributes $M^t \subseteq \mathcal{A}^t$ analyzed in the graph cube. □

The set of topological dimensions and measures form the set of topological attributes: $D^t \cup M^t = \mathcal{A}^t$. The particularity of topological measures is that they:

- require the graph structure to be computed.

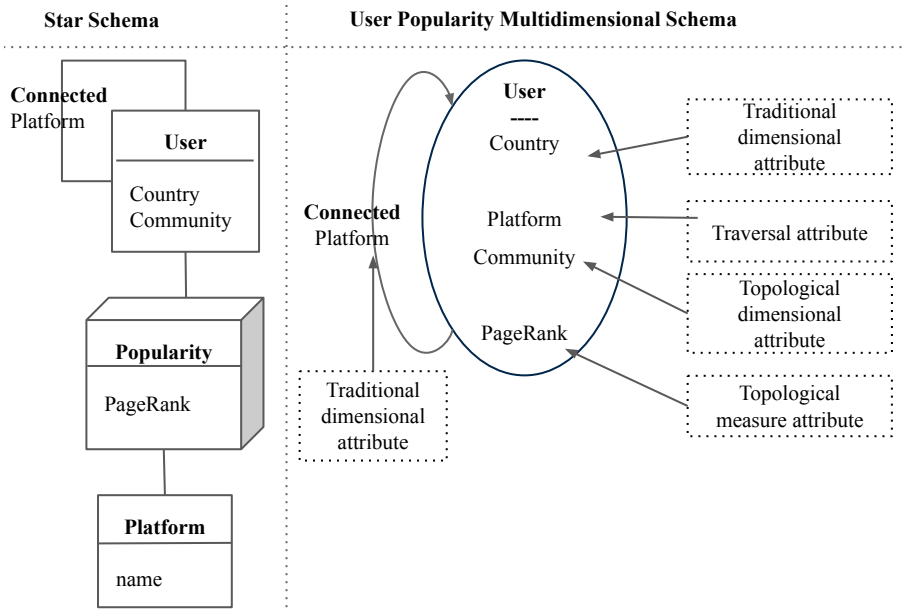

Fig. 5: Mapping Between OLAP Cube and a Multidimensional Topological Graph Schema

- are holistic, thus, they need to be recomputed after each aggregation, i.e., the graph algorithm to compute the topological attributes has to be executed after each graph aggregation, instead of applying traditional aggregation functions. The topological dimensions, however, need the graph structure only at the initial phase to compute the base graph cuboid.
- may need to be computed using a homogeneous subgraph of the multidimensional graph (most algorithms to compute topological properties such as centrality only make sense on homogeneous subgraphs).

Given a multidimensional graph, topological graph cubes are derived to capture the topological properties of graphs and represent them with numerical values (in contrast to traditional content-based properties that do not capture the topological characteristics).

Definition 11 [**Topological Graph Cube**] A topological graph cube is a graph cube that captures the topological properties of graphs and represent them with numerical values. It is obtained by restructuring the topological multidimensional graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{D}, \mathcal{M})$ in all possible aggregations through the topological dimensions and/or by embedding and aggregating topological measures. That is, $\exists \mathcal{D}^t \subseteq \mathcal{D} \parallel \exists \mathcal{M}^t \subseteq \mathcal{M}$. □

The model we propose in this paper could be mapped to a star schema shown in Figure 5. If the cube has a topological measure, we note that in

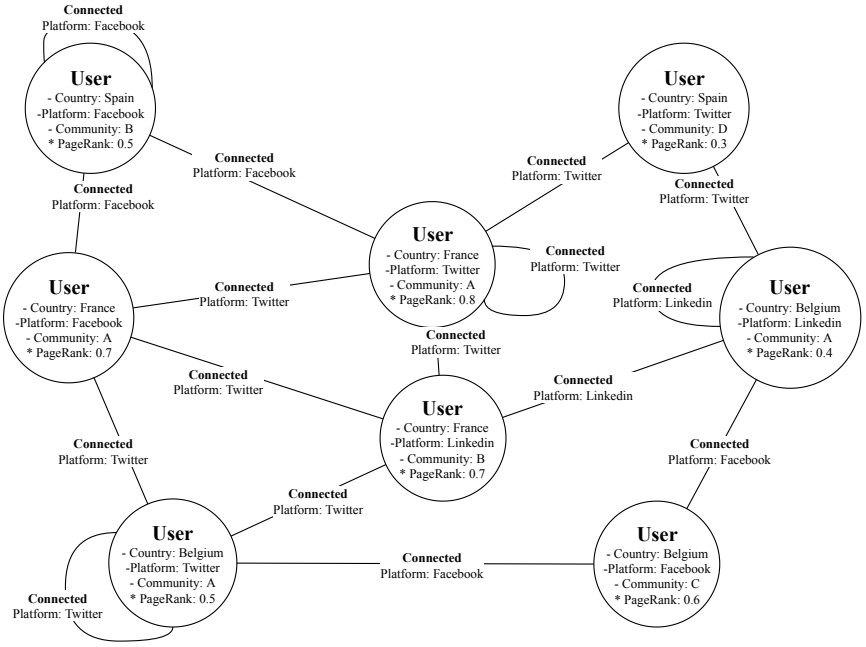

Fig. 6: Base Graph Cuboid Instance

order to guarantee a correct cube summarizability, roll up operations cannot be applied directly to OLAP cuboids to produce cuboids at higher aggregation levels, as typically done in OLAP. The reason is that following each roll up operation, the graph structure changes and so the topological values do. Specifically, given two cuboids C_i and C_j such that $C_i \subset C_j$, C_j results from aggregating C_i according to the multidimensional information stored in the topological multidimensional graph. However, the graph structure resulting cannot be expressed as a transitive function in terms of the input graph structure, as typically done in traditional OLAP cubes. Thus, topological graphs need to be computed once the cuboid they belong to has been created.

Example 3 (Popularity Graph Cubes) Given a property graph, we suggest a process to enrich the graph with topological attributes, and derive a potential multidimensional schema and later its corresponding OLAP cube. We consider the example of a property graph representing a social network as in Figure 4. A single type of nodes and a single type of edges are considered: $\mathcal{L} = \{User, Connected\}$. This graph is enriched to capture topological properties of users such as their community and PageRank. We design a possible multidimensional graph schema that embeds topological dimensions and measures. Three dimensions are considered: $\mathcal{D} = \{Country, Community, Platform\}$, and the analyzed measure is *PageRank*. The measure is computed using the PageRank algorithm that traverses each time the edges that have

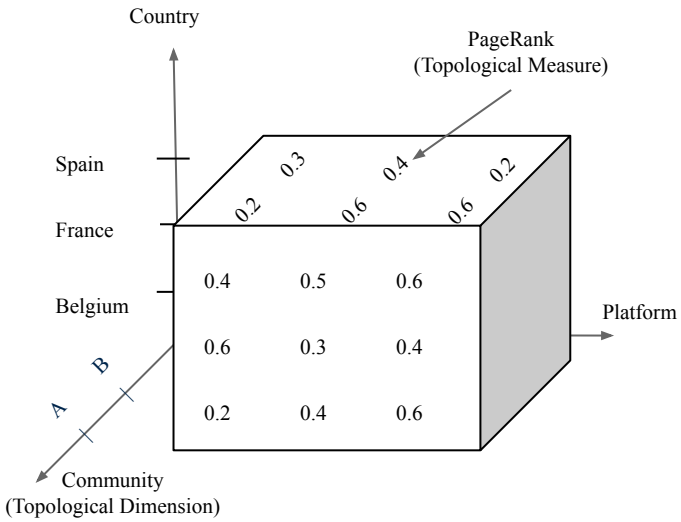

Fig. 7: Popularity Topological OLAP Cuboid

the same value on their platform attribute (the traversal attribute). Figure 5 shows the mapping between the multidimensional popularity graph and its corresponding star schema. An instance of the popularity multidimensional graph at the base level is shown in Figure 6. Figure 7 shows an example of the popularity OLAP Cube corresponding to the graph cuboid of Figure 6, where community, platform and country are the dimensions, and the PageRank is the topological measure. □

3.2 Topological Graph Cuboid Processing

In this section we describe the topological cuboid aggregation algorithm 1. This algorithm is used to build topological graph cuboids. It takes as input a topological multidimensional graph, perform its aggregation along the given dimensions, and then applies the chosen aggregation function to compute the new measure values. The main phases of the algorithm are the following (note that this algorithm guarantees the multidimensional integrity constraints discussed in Section 2.2):

1. Create a hash structure φ mapping each set of dimensional attributes from $\mathcal{D}'$ to an aggregate node/edge $u \in \mathcal{V}' \cup \mathcal{E}'$ (Line 2).
2. Create the set of aggregate nodes $\mathcal{V}'$: traverse the nodes of the multidimensional graph and create a node in $\mathcal{V}'$ corresponding to each subset of nodes in $\mathcal{V}$ sharing the same dimensional attribute values $\mathcal{D}'$ (Line 2-12). That is, for each node $u \in \mathcal{V}$ create its corresponding aggregate node $u' \in \mathcal{V}'$

Algorithm 1: Topological Cuboid Aggregation

Input :
- A topological multidimensional graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{D}, \mathcal{M})$
- Dimensions of the aggregate cuboid: $\mathcal{D}' \subset \mathcal{D}$

Output: An aggregate topological graph cuboid $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{L}, \mathcal{D}', \mathcal{M}')$

```
1  begin
2      Initialize a hash structure φ : D' → V' ∪ E'
3      for u ∈ V do
4          if φ(D'(u)) = NULL then
5              Create an aggregate node u' ∈ V'
6              L(u') ← L(u)
7              D'(u') ← D'(u)
8              M'(u') ← 0
9              φ(D'(u)) ← u'
10         for M'_i ∈ M' do
11             if M'_i is NOT topological then
12                 M'_i(u') ← compute(M'_i(u'), M_i(u))
13     for e(u, v) ∈ E do
14         u' ← φ(D'(u))
15         v' ← φ(D'(v))
16         if φ(D'(e)) = NULL then
17             Create an aggregate edge e'(u', v') ∈ E'
18             L(e') ← L(e)
19             D'(e') ← D'(e)
20             M'(e') ← 0
21             φ(D'(e)) ← e'
22         for M'_i ∈ M' do
23             if M'_i is NOT topological then
24                 M'_i(e') ← compute(M'_i(e'), M_i(e))
25     for M'_i ∈ M' do
26         if M'_i is topological then
27             for u' ∈ V' do
28                 M'_i(u') ← compute(u', M'_i, G')
29             for e' ∈ E' do
30                 M'_i(e') ← compute(e', M'_i, G')
31     return G' = (V', E', L, D', M')
```

(that has the dimensional attributes $\mathcal{D}'(u)$) if it was not already created in $\mathcal{V}'$ (Line 3-5). u and u' share the same label and dimensional attributes, and the measures are initialized (Line 6-8). The newly created node u' is stored as the value of the hash function corresponding to the dimensional attributes $\mathcal{D}'(u)$ (Line 9). Otherwise, if an aggregate node corresponding to u was already created, the non topological measure attributes $\mathcal{M}'(u')$ are updated using a user-defined function *compute* (Line 10-12).

3. Create the aggregate edges $\mathcal{E}'$: for each edge on $e(u, v) \in \mathcal{E}$, we retrieve the aggregate nodes in $u', v' \in \mathcal{V}'$ corresponding to its adjacent nodes $u, v \in \mathcal{V}$ (Line 13-15). If $e'(u', v')$ was not yet created, then a new edge $e'(u', v') \in \mathcal{E}'$

is created (Line 16-17). e and e' share the same label and dimensional attributes, and the measures are initialized (Line 18-20). The newly created edge e' is stored as the value of the hash function corresponding to the dimensional attributes $\mathcal{D}'(e)$ (Line 21). Otherwise, if an aggregate node corresponding to e was already created, the non topological measure attributes $\mathcal{M}'(e')$ are updated using a user-defined function *compute* (Line 22-24).

4. If a measure M_i is topological, then it needs the whole aggregate graph $\mathcal{G}'$ to be built. The topological values for the nodes and edges are computed using the aggregate graph $\mathcal{G}'$ by applying a user-defined function *compute* (Line 25-30). The computation of topological values involves usually iterative graph algorithms that traverse the whole graph $\mathcal{G}'$.

3.3 Deriving OLAP Cubes from Graph Cubes

In this section, we detail our approach to derive OLAP cubes from multidimensional graphs, and precisely from their corresponding graph cubes. Most of the state-of-the-art techniques focus either on building traditional OLAP cubes, or building graph cubes. Here we propose to establish the link between the two. Thus, designing OLAP cubes that leverage the content and the topology of the graph, and expose both numerical and graph-structured insights.

The main current assumption is that each graph cuboid can be loaded into a relational OLAP cube. This is true for content-based graph cubes. However, loading a graph cuboid into a relational cube causes the loss of the graph structure. This loss of the graph structure have a direct consequences for graph cube computation and analysis as follows:

- Graph cube computation: particularly roll up cannot be applied to get cuboids at higher lattice levels. The reason is that following each roll up operation, the graph structure changes. Therefore, the topological measures on the aggregate graph need to be recomputed.
- Graph cube analysis: operations such as subgraph matching and traversal could no longer be executed on the extracted OLAP Cube. Therefore, as discussed in the previous section, the ability to deal with topological graph cubes is lost.

Example 4 (Deriving Popularity OLAP cube from the Graph cube) Given the multidimensional graph model in Figure 4, we design a lattice, as shown in Figure 8. Each point in the lattice corresponds to a graph cuboid. For simplicity, we consider the dimension attributes: $\{Location, Community, Platform\}$, while ignoring the hierarchies of the location dimension. We highlight two particular aggregations: (1) node-only aggregations (i.e., only dimensional attributes from user nodes are kept not fully aggregated as in ($\langle Location, Community, * \rangle$, $\langle Location, *, *, * \rangle$, and $\langle *, Community, * \rangle$), and (2) edge-only aggregation as in ($\langle *, *, Platform \rangle$). The fact analyzed is the popularity of users. The measure is PageRank, computed by applying

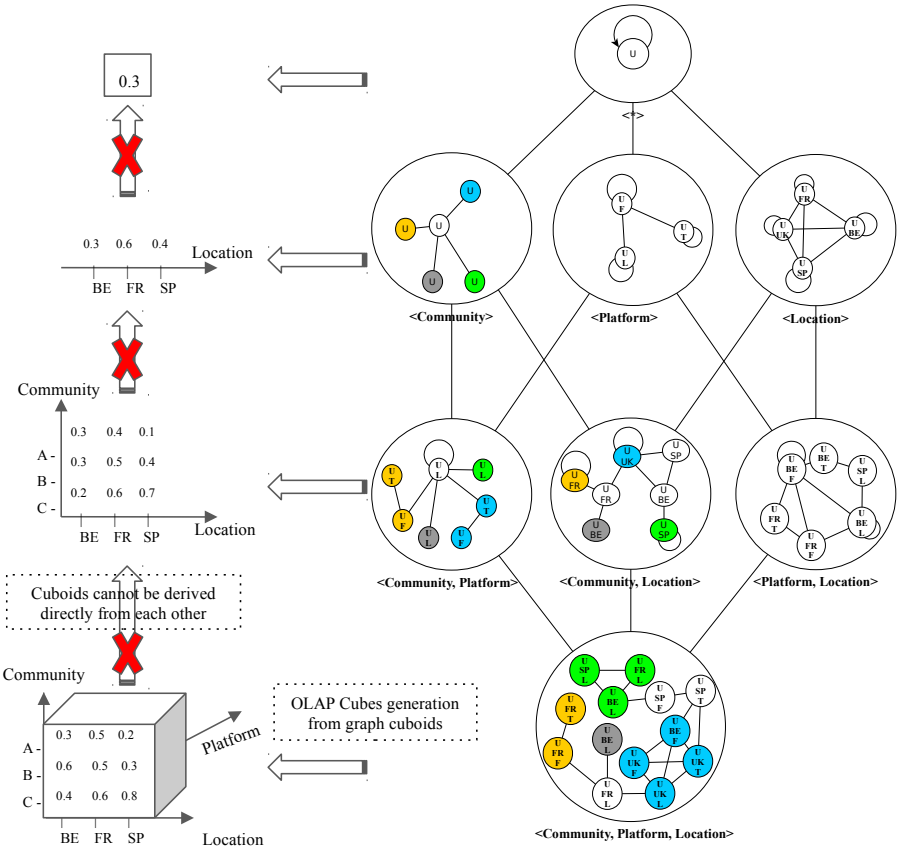

Fig. 8: OLAP Cube Generation form Graphs

the PageRank algorithm in the social network following the edges labeled *Connected*.
The figure depicts the coupled processes of (1) aggregation of graph cubes, and (2) generation of corresponding OLAP cubes and the mapping kept between them. This mapping is important, as the graph topology corresponding to each OLAP cuboid needs to be preserved in order to compute the topological measures such as PageRank. The measures could afterwards be loaded into the OLAP cubes for further multidimensional analysis. □

The mapping discussed in this section could help in the integration of graph data and graph analytics within current data warehouses. However, the link is pretty limited when graph analytics are combined with OLAP. This has been the main assumption behind current graph OLAP tools, but this is not realistic given the relevance of graph-specific algorithms when dealing with graph data. Thus, given that most graph-derived cubes could not be

supported with current relational warehousing systems, this motivates the need for building specialized graph OLAP warehousing systems.

4 Graph-structured Cubes

4.1 Graphs-structured Cube Model

Graph-structured cubes extend the traditional OLAP cubes with the capability of having the dimension and measure values represented as graphs. Current warehousing systems are not designed to support this type of cubes, which further motivates the need for developing native graph warehousing systems. In this section we formally define the concepts of graph-structured dimensions, measures and cubes.

The graph-structured dimensions are dimensions whose values are represented as graphs. They express complex dimension values that could not be represented by a simple value. This enables structuring the multidimensional space in a novel way capturing graph elements that are connected in a complex manner. Graph-structured dimensions provide therefore a powerful selection mean to examine the behaviour of non-trivial grouping of nodes or edges.

The definition of graph-structured dimensions relies on graph patterns defined as follows:

Definition 12 [**Graph Pattern**] A graph pattern $\mathcal{P}$, over a property graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$, is defined as $\mathcal{P} = (V_p, E_p, \alpha, \beta)$, where:

- V_p is a finite set of nodes.
- E_p is a finite set of edges.
- β is the predicate applied on the labels $l_i \in \mathcal{L}$ of the graph elements. This predicate is a conjunction of atomic predicates used to compare the label specified on the pattern with the actual label of the node (resp. edge). Given a label l_i and a string s_i, the comparison is of the form l_i **op** s_i, and is performed using one of the two equality comparison operators $=, \neq$.
- α is the predicate applied on the attributes $A_i \in \mathcal{A}$. This predicate is a conjunction of atomic predicates that each of them compares a constant c specified on the pattern with the value of the attributes on a given graph elements (e.g., $A_i(v)$). The comparison is done using one of the following comparison operators: $<, \leq, =, \geq, >, \neq$. □

Definition 13 [**Graph-structured dimension**] A graph-structured dimension D_i is a dimension represented with a graph pattern $\mathcal{P}$ used for selecting a subset of the graph elements. The set of graph-structured dimensions is $\mathcal{D}^s = \{{D_1}^s, {D_2}^s, ..., {D_n}^s\} \subseteq \mathcal{D}$. Each graph-structured dimension ${D_i}^s$ is represented by a graph pattern $\mathcal{P}_i$. □

The graph-structured measures are measures where the values are represented as graphs, which enables capturing and exposing insights and metrics structured as graphs. Another main benefit of graph-structured measures is

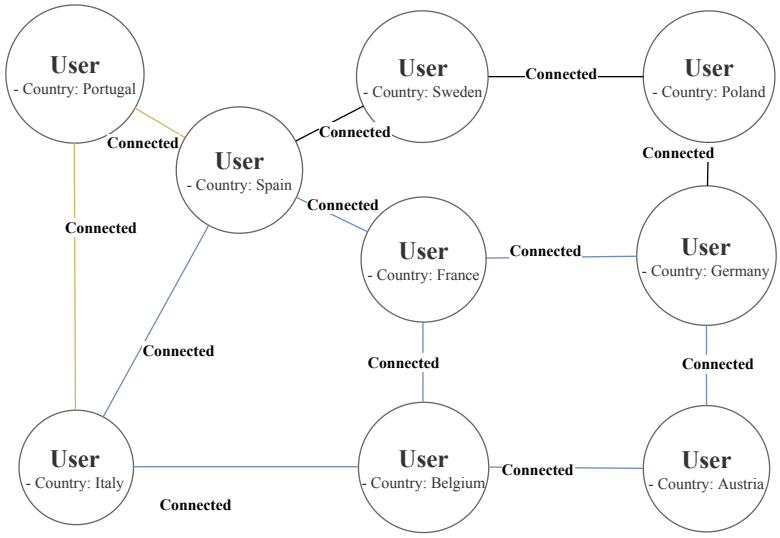

Fig. 9: Graph-structured Cuboid

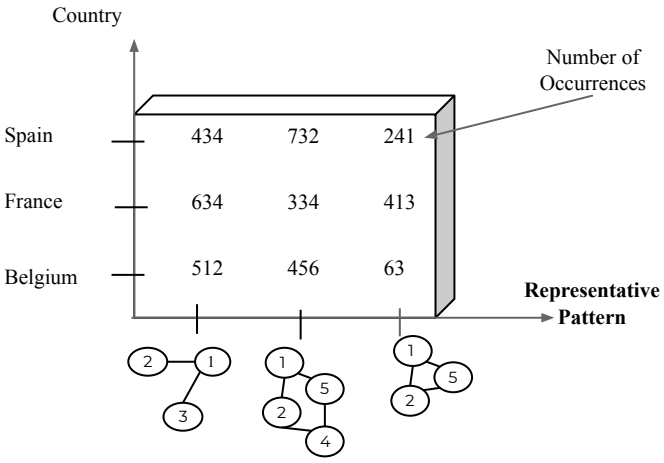

Fig. 10: Graph-structured Dimension

that they minimize the information loss, as they keep the graph structure after being computed or aggregated.

Definition 14 [**Graph-structured Measure**] A graph-structured measure $M_i{}^s$ is represented with a subgraph $\mathcal{G}^s$. It is computed using a graph function Δ that takes a graph as input and returns a graph, such as most frequent pat-

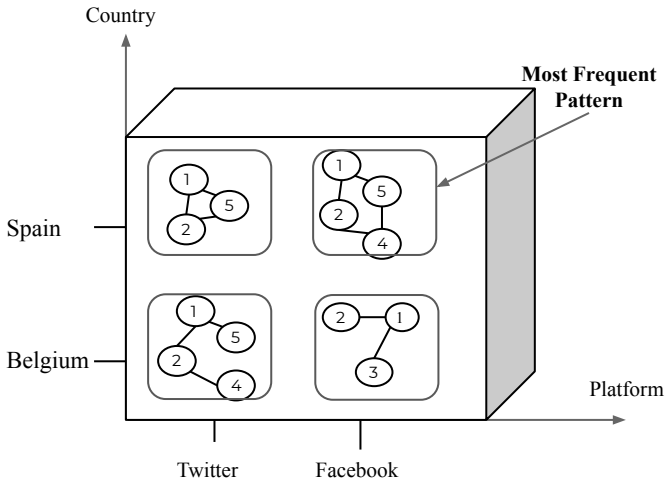

Fig. 11: Graph-structured Measure

tern, or minimum spanning tree. An aggregation function $\mathcal{F}$ is used to compute the graph-structured measure at different aggregation levels such as intersection or union of graphs. $\mathcal{F}$ and Δ could be the same function (thereby recomputing the measure after each aggregation). $\mathcal{M}^s = \{{M_1}^s, {M_2}^s, ..., {M_n}^s\} \subseteq \mathcal{M}$ □

Definition 15 [**Graph-structured Cube**] A graph-structured cube is a graph cube that captures and represents the dimensional concepts using graphs. Therefore, it contains either graph-structured dimensions or graph-structured measures, or both. A graph-structured cube is obtained by restructuring the multidimensional graph in all possible aggregations through the graph-structured dimensions and/or by embedding and aggregating graph-structured measures. □

In the following example, we illustrate two graph-structured cubes, highlighting respectively graph-structured dimensions and measures. For each cube, we show how graph-structured multidimensional structures could be combined with the numerical ones defined in the previous sections.

Example 5 (Graph-structured Cubes) Consider the graph cuboid of Figure 9. It represents a graph cuboid where users are grouped using the country dimension. Figure 10 shows a cube that highlights the case of graph-structured dimensions. For this example, we assume that we can extract a set of patterns that represent the graph elements, and we call these patterns the representative patterns. The horizontal axis of the cube is then populated by a set of graph patterns depicting the representative patterns. Using this cube, the user can analyze for example how often users from a given country are involved in

a representative pattern . Those are complex dimension values that could not be represented by a numerical value, and need therefore to be defined by patterns. To define the dimension values, the user could either find the pattern using graph algorithms, or use his domain knowledge. The three patterns of the graph-structured dimension of the cube of Figure 10 are represented on the graph cuboid Figure 9 using different colors for each. Figure 11, on the other hand, puts the focus on graph-structured measures. The measure studied here is the most frequent pattern, which could be obtained by applying the graph algorithms. Each measure (i.e., frequent pattern) is then placed within the graph-structured cube using two traditional dimensions: country and platform. This cube could be used to analyze the most frequent pattern observed by country and platform. □

4.2 Graph-structured Cuboid Processing

Graph-structured cuboid aggregation is similar to topological cuboids' aggregation. It is performed along the lines of Algorithm 1, while adapting the selection and aggregation to handle the graph patterns. It takes as input a multidimensional graph and performs its aggregation along a given set of dimensions, then applies a chosen graph aggregation function to compute the new measure values. Considering a cube with graph structured dimension and measures, the main steps of the algorithm are:

1. Create a hash structure $\varphi : \mathcal{P}' \rightarrow \mathcal{V}' \cup \mathcal{E}'$, mapping each pattern (corresponding to a graph-structured dimension) to an aggregate node/edge.
2. Create the set of aggregate nodes $\mathcal{V}'$: traverse the nodes of the multidimensional graph and create a node in $u'_i \in \mathcal{V}'$ corresponding to the subset of nodes in $G_u = \{u_1, ..., u_n\} \subseteq \mathcal{V}$. G_u is the set matching the pattern P'_i representing a dimensional value of D'_i. The aggregate node $u'_i \in \mathcal{V}'$ is only created if no nodes in $\mathcal{V}'$ corresponding to P'_i was already created. u and u' share the same label and non-structured dimensional attributes, and it gets assigned its graph-structured dimension D'_i. The newly created node u' is stored as the value of the hash function corresponding to the pattern P'_i. Otherwise, if an aggregate node corresponding to u was already created, the non topological measure attributes $\mathcal{M}'(u')$ are updated.
3. Create the aggregate edges $\mathcal{E}'$: this step is similar to the one in Algorithm 1. That is, for each edge on $e(u, v) \in \mathcal{E}$, if no corresponding edge $e'(u', v') \in \mathcal{E}'$ was created, e' is created and inherits the same label and dimensional attributes of e. The aggregate edge e' is stored as the value of the hash function corresponding to P'_i. Otherwise, the non topological measure attributes $\mathcal{M}'(e')$ are updated.
4. Topological and graph-structured measures are computed for the aggregate graph.

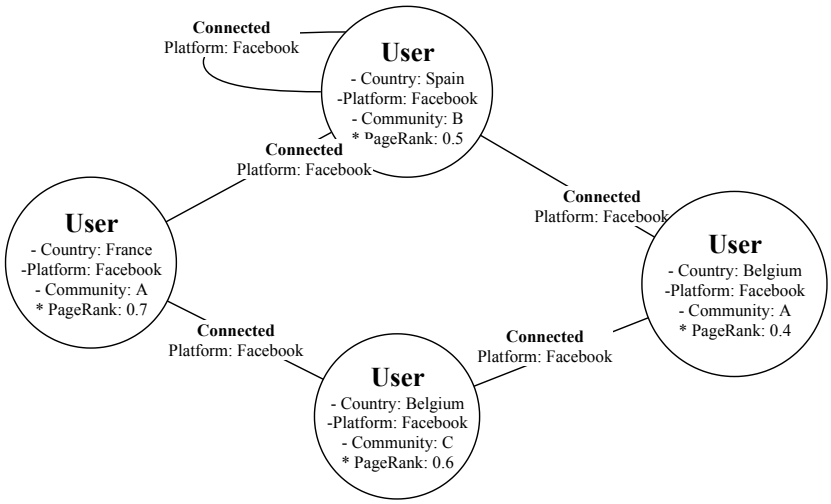

Fig. 12: Slice on the Facebook Platform Dimension

5 OLAP Analysis of Graph Cubes

OLAP analytics supports interactive and complex queries over large volume of data, from different perspectives and through different hierarchical levels. They, enabling analysts to highlight the data item of interest, and then drill down to the underlying data from which it has been created. This could help in decision support scenarios such as the measurement or comparison of the business performance across the different dimensions. In this section, we describe a set of algebraic operators for OLAP querying of multidimensional graphs. We consider the graph cubes defined and computed in the previous sections as the fundamental construct of the multidimensional model. The graph cubes are the operand and the return type of all OLAP operations. We illustrate the application of each operation on a graph cuboid and on its corresponding OLAP cube. In addition to cuboid and crossboid operations that were defined in the literature (Zhao et al., 2011), we present the major OLAP operations applied on graph and OLAP cubes.

Multidimensional Selection Multidimensional selection (also called a slice) (denoted as $\sigma_{\mathcal{P}}(\mathcal{G})$) restricts the graph $\mathcal{G}$ to a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ where all nodes and edges match the selection pattern $\mathcal{P}$. The selection pattern could be a conjunction of (1) atomic predicates applied to one or more dimension attributes in the case of property and topological cubes, or (2) graph patterns in the case of graph-structured cubes, or (3) a combination of both. The result $\mathcal{G}'$ is a set of nodes and edges that are matched by the selection pattern $\mathcal{P}$. The algebra of the selection operator is defined as follows:

– Input: A graph cuboid $\mathcal{G}$ and a selection pattern $\mathcal{P}$.

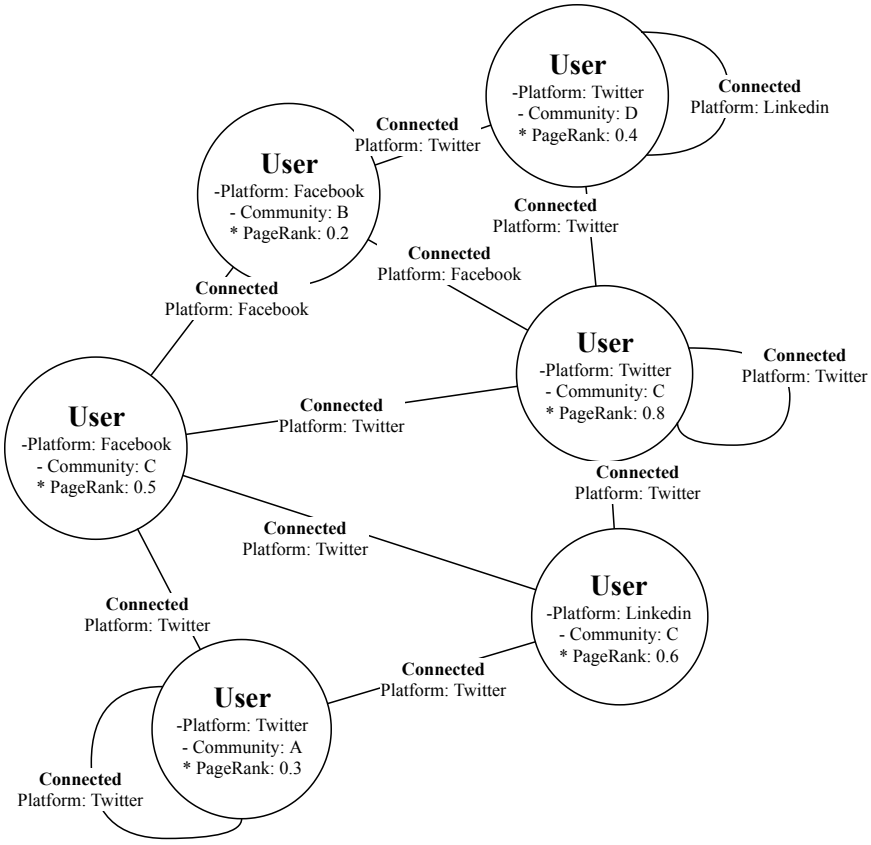

Fig. 13: Popularity Graph Cuboid Rolled up to $\langle *, Community, Platform \rangle$

- Output: A graph cuboid $\mathcal{G}' \subseteq \mathcal{G}$, that matches the selection pattern $\mathcal{P}$.
- Example: The result of a selection applied on the graph cuboid of Figure 6 is shown on Figure 12, where only user nodes of the Facebook platform are selected.

Roll-up and Drill-down Roll-up (denoted as $\mathcal{R}_{D_i}(\mathcal{G})$) aggregates the graph $\mathcal{G}$ along the dimension D_i. The graph is either aggregated to the next dimension hierarchy level if D_i is part of a dimension hierarchy following the partial order $\mathcal{R}$, or to ALL. This operation modifies the granularity of the graph by means of a many-to-one relationship which relates instances of two levels in the same dimension hierarchy, corresponding to a part-whole relationship.
This operation performs structural changes to the graph, and generates a new graph placed at the next level of the dimension hierarchy, while respecting the summarizability integrity constraints. Roll-up is implemented in three phases (1) first a selection of graph elements matching the aggregation pattern $\mathcal{P}_{agg}$

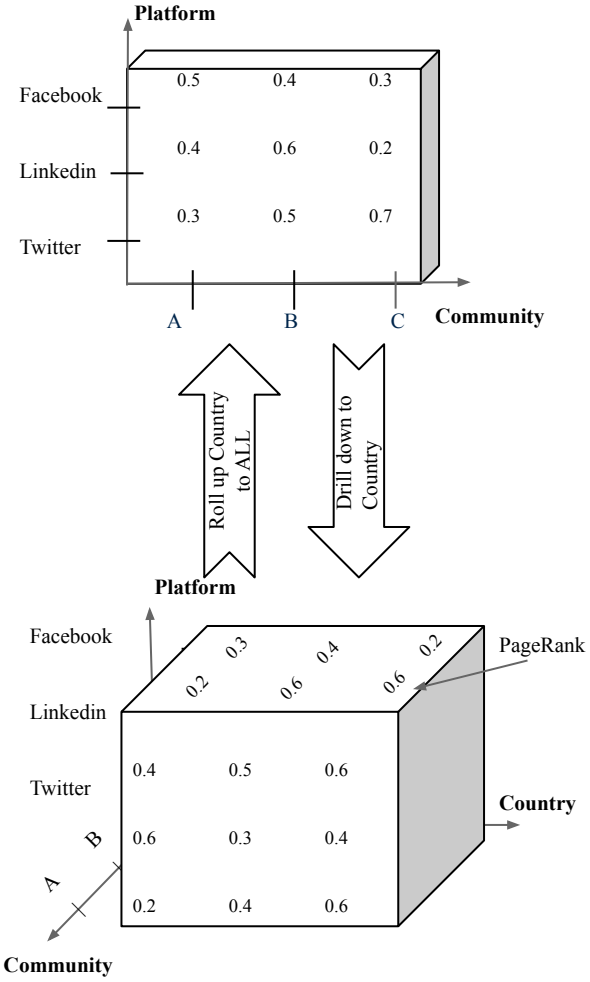

Fig. 14: Roll-up and Drill-down on the Popularity OLAP Cube

that describes the graph elements at $Level_i$, then, (2) the graph aggregation to shape the graph at $Level_{i+1}$, and finally (3) measures are (re)computed and placed on the aggregate graph. The algebra of the roll-up operator is defined as follows:

- Input: Initial graph cuboid: $\mathcal{G}$; The dimension to aggregate D_i.
- Output: A graph cuboid $\mathcal{G}'$. All elements of the initial graph cuboid $\mathcal{G}$ that contain the dimensional attributes of D_i are grouped in their corresponding node (resp. edge). The measure values on the aggregate nodes and edges are computed according to their aggregation function $\mathcal{F}$.

– Example: The result of a roll-up $Country \rightarrow ALL$ applied on the graph cuboid of Figure 6 is shown on Figure 13, where user nodes are grouped by community and platform, and the new page rank value is computed for each node. Figure 14 shows the equivalent roll-up and drill down operations applied on the corresponding OLAP cubes.

Roll-up is similar to the *cuboid* operation defined in the GraphCube paper (Zhao et al., 2011). Drill-down is the inverse of roll-up, and can only be applied if we previously performed a roll-up and did not lose the correspondences between the graphs.

Drill-across and Projection This operation changes the subject of analysis of the cube by means of a one-to-one relationship. The n-dimensional space remains exactly the same, only the cells placed on it change. With this operation, different measures are placed on the same multidimensional space. This operation translates to a join between two graph cuboids put on the same multidimensional space, at the same aggregation level. The join condition for nodes is their identifiers. Projection is the reverse operation of a drill-across. It selects a subset of measures of interest to be studied within the multidimensional space. The algebra of the drill-across operator is defined as follows:

– Input: Initial graph cuboids: G^1, G^2, and the measures m_i, m_j.
– Output: A graph cuboid G^3, union of G^1, G^2, where the concerned graph elements embed both measures m_i and m_j.
– Example: Figure 14 shows an example of a drill-across between a topological and a graph-structured cube. Both cubes are placed in a cube having as dimensions $(\langle Community \rangle, \langle Country \rangle)$. The first is a graph-structured cube containing representative communities, and the second is a topological cube containing PageRank. Using drill-across, the measures from the two cubes could be embedded in the same cells and analyzed within the same cube. Inversely, a projection would for instance remove the measure representative community from the cube to focus only on studying the PageRank.

In the same way, further OLAP operators could be applied on the graph cubes for richer or more intuitive analysis. For example, the difference between graphs removes isomorphic subgraphs that exist in the two input graphs. Drill-through enables direct access to the subgraph that was initially used for the computation of the cube's measures. It goes beyond drill-down to explore the lowest aggregation level present in the physical graph, and non-necessarily reached at the data mart level. In general, this paper opens the door to advanced operators combining graph-like and OLAP operators.

6 Implementation and Experiments

Current decision-support systems, and particularly data warehouses, were designed to support relational data management and analysis. Due to the fundamental difference between graph and relational data, the existing systems are

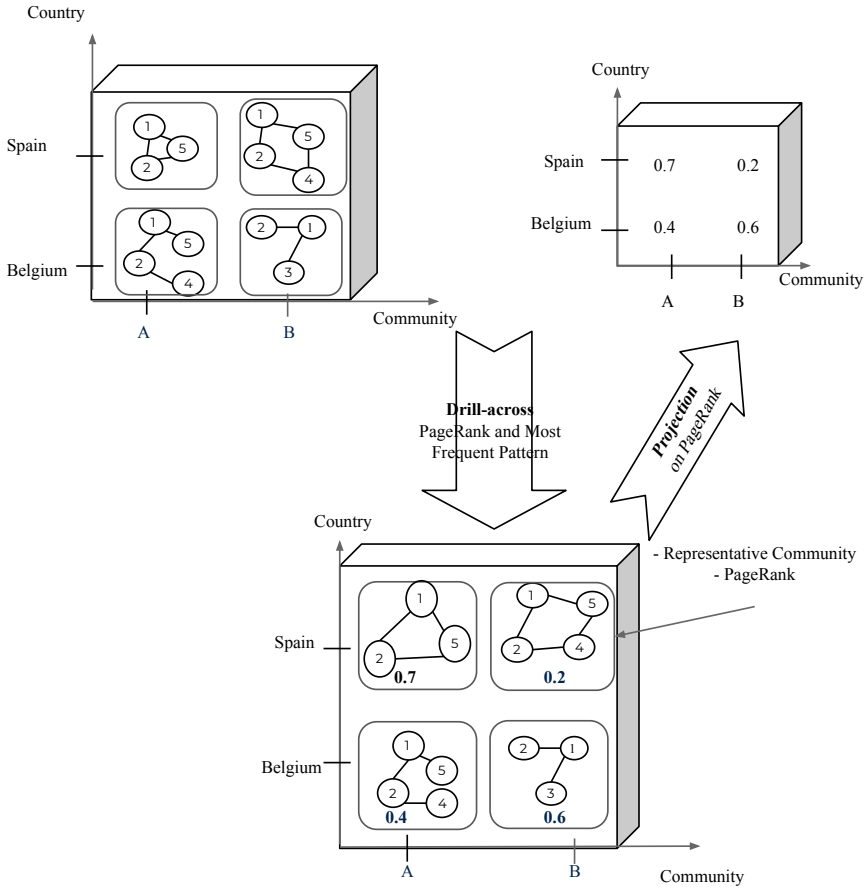

Fig. 15: Drill-across and Projection

not suitable for efficient graph analysis. The structure-driven management and analytics of graph data call for rethinking the architecture of data warehouses to support graph analytics, and to the development of novel data models, query processing paradigms and storage techniques.

6.1 Framework Architecture and Implementation

The architecture of the graph warehousing and analysis framework, is depicted in Figure 16. To exemplify it, we use the same running example and take Twitter as source data. The modules are described as follows:

– Graph Extraction: Graph data is extracted from the source. In our running example, by means of the Twitter streaming API. A set of transformations is then applied in order to cleanse the data and fit it within the envisioned

schema. The stream is parsed to identify the data entities and merge duplicates, and compute new attributes such as length of tweets and their sentiment. For this purpose, any generic tool would suffice.
- Graph Construction: The clean data is loaded in the graph store. In this case, we used Neo4j to store the graph data and Cypher queries to perform the loading. The cleansed and integrated Twitter data is therefore natively stored and managed as a multidimensional graph.
- Graph Cube Construction: Multiple multidimensional schemas could be built from the same graph warehouse to satisfy the various analysis needs. The semantic relativism inherent in graphs allows creating several views from the same data and making them co-exist in a much simpler way than any other data model. Therefore, given a graph lattice, the graph cube framework enables the computation and the aggregation of the corresponding graph cuboids. Each graph cuboid is computed, and persisted in a graph store that resembles a graph mart. The graph cuboid stores natively different graph measures (e.g., centrality, shortest paths, frequent patterns etc.). An example of the graph cuboid computation is shown below.
- Graph Analysis: Complex and interactive analysis of graph cubes is performed at this phase. In contrast to traditional OLAP analytics, graph analytics enables BI-oriented analysis of graph metrics stored in the graph cuboids. For example, analysts could examine at different levels of aggregation and from multiple perspectives graph measures such as influence (e.g., computed using centrality), or identifying communities and their connections (e.g., computed using graph clustering). Importantly, note that traditional visualisation tools do not suffice to deal with interactive graph analysis, especially graph-structured cubes. Therefore, an ad-hoc graph browser was implemented.

We implemented the architecture described above as a prototype graph warehousing system. We used Neo4j for the graph data management and Neo4j graph algorithms and JUNG (Java Universal Network/Graph Framework) for the graph mining. The Java code below shows an example of cuboid computation to perform an aggregation on the dimensional attribute: *sentiment*. Here, we specify the input dataset and output directory, the dimensions and their dimensional attributes, the measures and their computation and aggregation functions. The following code illustrates our API:

```
1  //Graph Cuboid Builder
2  GraphAggregator graphAggregator = GraphAggregator.builder()
3  //Input: Multidimensional Graph
4  .basePath(Paths.get("data/MDTwitter"))
5  //Output: Graph Cuboids
6  .workPath(Paths.get("data/TwitterCuboid"))
7  //Vertices
8  .vertex("User")
9  //Vertex dimensions
10 .dimension("Date", DimensionAgg.KEEP)
11 .dimension("Language", DimensionAgg.KEEP)
12 //Traditional vertex measure
```

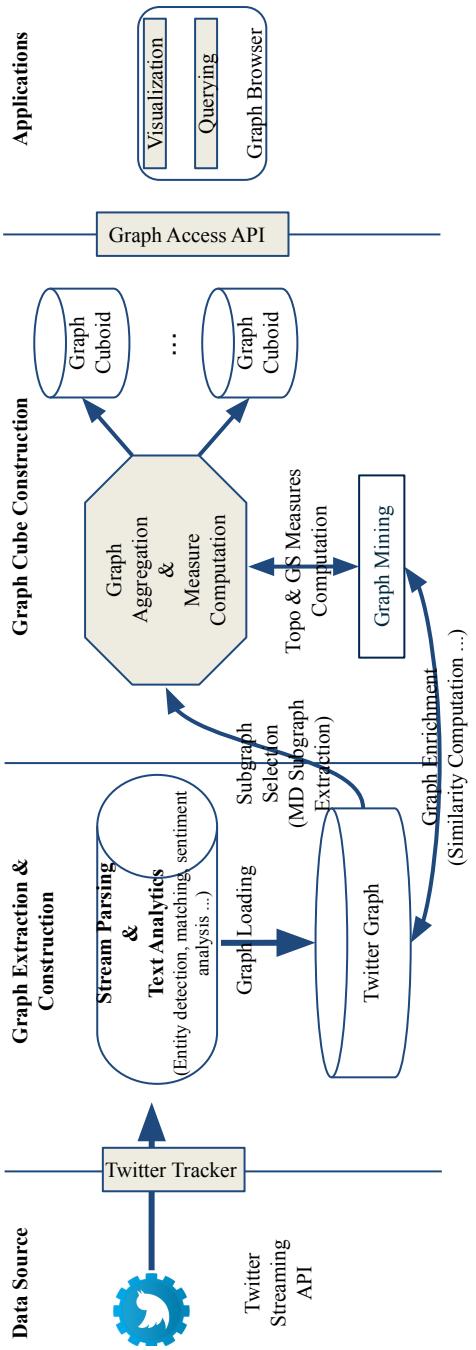

Fig. 16: Twitter Network Warehousing Architecture

```
13  .measure("followers", "TotalFollowers",
14  AggFunction.COUNT)
15  .vertex("Tweet")
16  //Vertex dimensions
17  .dimension("Language", DimensionAgg.KEEP)
18  .dimension("Sentiment", DimensionAgg.IGNORE)
19  //Topological vertex measure
20  .structuralMeasure(StructuralMeasure.SM.LOUVAIN,
21  "RETWEETED")
22  //Edges
23  .edge("POSTED")
24  .edge("MENTIONED")
25  .edge("REPLIED_TO")
26  .edge("RETWEETED")
27  .build();
28  graphAggregator.aggregate();
```

6.2 Experiments

In this section, we present the experimental results of our graph OLAP framework using multiple real-world datasets. We compare the cuboid generation and aggregation time for each dataset at different aggregation levels.

Datasets We ran the experiments on two types of real world datasets. The first are three Twitter datasets, of size 500K, 1M and 2M edges. The data is collected using Twitter streaming API as depicted by the framework of Figure 16 described above. The original stream contained two types of nodes: *User and Tweet*, and four types of edges: *POSTED, RETWEETED, MENTIONED, and REPLIED_TO*. We enriched the Tweet nodes by computing the sentiment of the tweets. Table 1 provides a summary of the characteristics of the multidimensional social network built using the Twitter datasets. The code to build Twitter cuboids was shown in the previous subsection.

	Dimensional Attributes	Measures
User	Language, Subscription Date	Number of Followers, Number of persons
Tweet	Language, Sentiment	Number of tweets, Community
Edges	none	Number of edges

Table 1: Twitter Datasets

The second type of graphs uses four datasets from the SNAP collection (Leskovec & Krevl, 2014). The original dataset contains only the graph structure between users. We use this dataset to experiment the computation and aggregation of topological dimensions and measures of the multidimensional graph. For the nodes, we computed three topological properties that we considered as dimensions (PageRank, triangles and clustering coefficient). For the measures, we computed community by label propagation ad considered it as a

measure for the nodes. We consider the count of nodes and edges as a measure each time we aggregate the graph. The ability to derive new multidimensional measures and dimensions using only the graph structure shows an interesting aspect of graphs and the potential of multidimensional graph analytics, even when we have no content related information about the original data. Table 2 shows the evolution of the graph order and size through consecutive multidimensional aggregations of the graph. Since the graph is homogeneous, we end up always with a single node and edge that summarizes the graph at the apex level.

Dataset	Original		Base		TR-PR		PR		Apex	
	#V	#E	#V	#E	#V	#E	#V	#E	#V	#E
DBLP	317,080	1,049,866	40,598	869,814	13,926	741,466	275	16,552	1	1
Youtube	1,134,890	2,987,624	41,704	1,835,365	15,067	1,459,786	652	51,840	1	1
Skitter	1,696,415	11,095,298	164,462	7,865,009	55,430	5,994,331	1,176	119,731	1	1
LiveJournal	3,997,962	34,681,189	564,648	33,382,661	122,331	28,683,757	700	70,480	1	1

Table 2: Graph Cuboids Order and Size

To build the graph cuboid for the SNAP graphs, we use the following code:

```
1  //Graph Cuboid Builder
2  graphAggregator = GraphAggregator.builder()
3  .basePath(Paths.get(DB_PATH))
4  .workPath(Paths.get(DB_PATH + "_aggregatedbase"))
5  .vertex(node)
6  .dimension("pagerank", DimensionAggregation.KEEP)
7  .dimension("coefficient", DimensionAggregation.KEEP)
8  .dimension("triangles", DimensionAggregation.KEEP)
9  .structuralMeasure(StructuralMeasure.SM.LABEL_PROP, edge)
10 .edge(edge)
11 .build();
```

Framework Efficiency The graph extraction and construction algorithms and the experimental setup were implemented in Java. For the first type of datasets, the framework was tested on a single machine with 16 GB of RAM, and an Intel(R) Core(TM) i5-7200U CPU@2.50GHz, running on Ubuntu 18.04. For the second type, with larger datasets, we used a machine with 256 GB of RAM, and an Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz, running on Ubuntu 18.04. The proposed system uses the centralized graph database Neo4j. The graphs were implemented using adjacency lists as it is a more compact representation. For the processing, hashmaps are used as described in the algorithm.

Given that the Twitter dataset is a heterogeneous graph, with multiple types of nodes and edges, we compute the time to build the cuboid at the base and apex level, and at the end two aggregate cuboids that aggregate the tweets(Tweet-Agg) and users(User-Agg) respectively. For the base level, Base-C refers to cuboid computation with content measures only, while Base-T refers to cuboid that has both topological and content measures. Figure 17

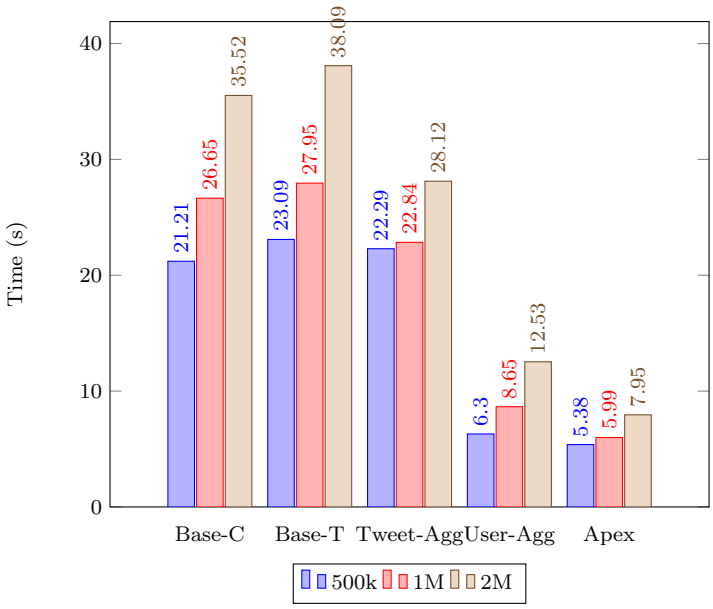

Fig. 17: Computation Time for Building the Graph Cuboids

shows the computation time with the JVM Xms and Xmx set to 8 GB. Figure 18 shows the aggregation of the SNAP networks. Given the raw datasets, first the multidimensional graph is computed, then aggregated through different dimensional levels. We run the experiment on the machine with 256 GB, but we set the JVM Xms and Xmx to 32 GB, except for the livejournal aggregations where we encounter an out of memory error. The results on Figure 18 show the computation time for the different graph cuboids. MD refers to the computation of the multidimensional graph, given the raw input from SNAP. Base refers to the base graph cuboid, PR-TR is the cuboid where the coefficient is aggregated, and PR is the cuboid where the graph is aggregated on both coefficient and triangles dimensional attributes, and Apex to the highest aggregation level.

Following these experiments, we notice that the processing time depends on the size, order and volume of the input and output graphs. The order of the graph is the number of its nodes and size refers to the number of its edges. As we consider both content and structural information present in the graph, all these properties have a direct effect on the efficiency of the aggregation. Given that at the first aggregation level we have many possible dimension values for the dimensional attributes, we end up with a graph close in size and order to the base graph, therefore exhibiting similar computation time. This explains, for example, why User-agg that aggregates users is faster than Twitter-agg. We also notice that most of the computation time is spent on the two phases: the I/O phase, where the graph is loaded from and to the disk, then the graph

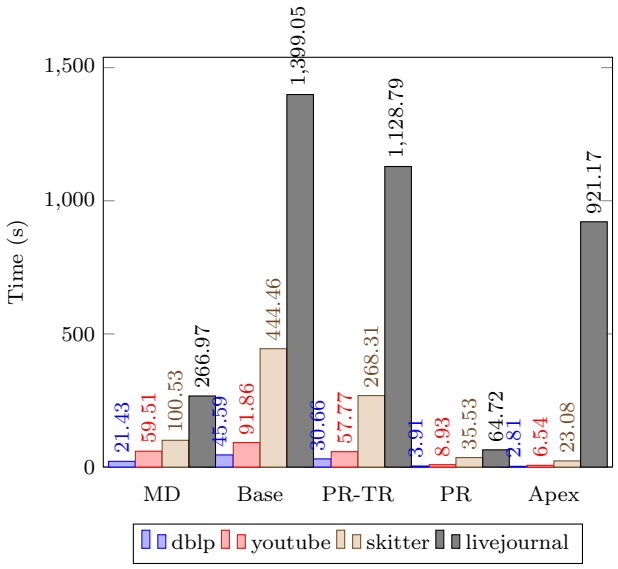

Fig. 18: Cuboid Aggregation Time

aggregation phase, where the nodes and edges are merged. The overhead of computing topological measures is very small as shown in Figure 17. This is due in part to the fact that the graph algorithms are executed within the database engine, without loosing I/O to export the graph to a processing library then import it again. Therefore, we got performances order of magnitude better than those when we used an external generic Java graph library such as JUNG and jGraphT.

7 Related Work

Graph Warehousing. A lot of research has been devoted for extending data warehousing and OLAP technology beyond the relational systems (Cuzzocrea, Bellatreche, & Song, 2013; Cuzzocrea, Saccà, & Ullman, 2013). Various efforts were led to support other data formats such as text (Lin, Ding, Han, Zhu, & Zhao, 2008), multimedia (Jin et al., 2010), and graphs (Queiroz-Sousa & Salgado, 2019). Multiple architectures and systems were proposed in the literature to integrate graph data into business intelligence systems. BIIIG (Petermann, Junghanns, Müller, & Rahm, 2014) is a framework for business intelligence on graphs that focuses on the use of the graph's flexibility in data integration. It enables integrating and referencing heterogeneous data from different sources. Li et al. (Li, Yu, Zhao, Xie, & Lin, 2011), proposed conceptual models for designing and querying graph data warehouse systems. In (Skhiri & Jouili, 2013; Ghrab et al., 2018), authors suggested novel architecture for graph BI systems that leverages large graph mining and warehousing. This paper goes

in-line with these research directions, and attempts to provide a foundation for extending decision-making systems, and particularly OLAP, with graph analytics capabilities, while paying particular attention to the few cases of possible correspondence between graph and ROLAP cubes.

Graph OLAP. Early research in graph warehousing started with the Graph OLAP model, which set the first foundations for multidimensional modeling and analysis of graphs. Graph OLAP supports the multidimensional modeling and analysis over a collection of homogeneous graph snapshots (Chen, Yan, Zhu, Han, & Yu, 2009). Two types of modeling and analysis are performed: (1) informational and (2) topological. In informational OLAP (I-OLAP), the dimensions are attributes of the graph snapshot. The aggregation of the graph is performed by overlaying and merging a set of graph snapshots that share the same dimension values. The analysis consists in edge-centric snapshot overlaying. Thus, only the edges are merged and changed, with no changes made to the nodes. In topological OLAP (T-OLAP), the attributes of the nodes are called topological dimensions. The aggregation consists of merging nodes and edges by navigating through the nodes' hierarchy. T-OLAP was discussed in a more detailed framework for topological OLAP analysis of graphs (Qu et al., 2011). The paper discussed the topological aggregation of the graph following the OLAP paradigm. They presented techniques based on the properties of the graph measures (T-Distributiveness and T-Monotonicity) for optimizing measures computations through the different aggregation levels. Another multidimensional model (Berlingerio, Coscia, Giannotti, Monreale, & Pedreschi, 2013), similar to Graph OLAP, was proposed ad considered the dimensions as the labels of the edges, and presented a set of analytical graph-based measures relevant for OLAP analysis of graph data. HMGraph introduced a data warehousing model for heterogeneous graphs focusing on edge-based dimensions (Yin, Wu, & Zeng, 2012). It enriched the informational and topological dimensions with the entity dimension and the rotate and stretch operations along with the notion of metapath to extract subgraphs based on edges traversals.

Graph Cube. The second family of frameworks focused on the efficient computation and extending the querying of OLAP cubes derived from multidimensional graphs. (Zhao et al., 2011) introduced the first framework that coined the term GraphCube. The authors defined a multidimensional graph from a single, homogeneous attributed graph, by choosing a subset of the attributes of the nodes to be the dimensions. The aggregate graph itself is the measure. The graph cube is obtained by restructuring the initial graph in all possible aggregation. The framework introduced two types of queries: (1) the cuboid query, which generates 2^n aggregate graphs, and (2) the crossboid query, which analyze the interrelationships between different graph cuboids. Many frameworks were proposed afterwards to (1) support more general graph models, (2) new types of multidimensional structures, (3) novel OLAP queries, and (4) custom materialization strategies. Pagrol introduced a parallel graph cube

framework that extended the original GraphCube model by defining the Hyper Graph Cube model that considers the attributes of the nodes and edges as dimensions (Wang et al., 2014). Both GraphCube and Pagrol designed various materialization policies to speed up the computation and analysis of graph cubes. However, both GraphCube and Pagrol were still limited to homogeneous graphs. The graph model was later extended with a framework for building OLAP cubes supporting heterogeneous attributed graphs and dimension hierarchies (Ghrab, Romero, Skhiri, Vaisman, & Zimányi, 2015). TSMH framework introduced the concept of relation path to guide the graph aggregation and building two new types of cubes: Entity Hyper Cube and Dimension Cube (Wang, Wu, & Wang, 2015). P&D Graph Cube extended the graph cube model by introducing the concept of path and dimension aggregate networks, along with their materialization strategies (Wu, Wu, & Wang, 2017). A multidimensional model for directed multi-hypergraphs and its query language were proposed in the literature, along with an implementation using Neo4j (Gómez, Kuijpers, & Vaisman, 2017). Other research lines focused on applying graph warehousing for specific domains such as the analysis of bibliographic data (Loudcher, Jakawat, Soriano-Morales, & Favre, 2015), or business process data (van der Aalst, 2013; Benatallah, Motahari-Nezhad, et al., 2015). For example, distributed OLAP analytics of process execution data represented as graphs was tackled by designing a Hadoop-based framework (Benatallah et al., 2015). Thus enabling, multi-level and multi-perspective analysis of large volumes of business process data represented as graphs.

Comparison. Existing work for OLAP analysis on graph provided a foundation for OLAP cubes computation and querying on graphs. Table 3 summarizes the related work from a modeling and implementation perspectives. As shown in the table, in most of state-of-the-art frameworks, the only measure that is examined is the aggregate graph itself, and the dimensions are a set of attributes or paths. **TopoGraph extended these frameworks by supporting the general case of property graph model and proposing new types of graph cubes that embed novel types of measures and dimensions. For these new graph cubes, the algebraic OLAP operators required for their analysis were defined and illustrated**. Further, to the best of our knowledge, this paper is the first to discuss the multidimensional integrity constraints on graphs. These constraints are a key concept that guarantees the correctness and soundness of graph cube construction aggregation. Moreover, for each of the introduced cubes, the possible correspondence with ROLAP cubes was discussed in an effort to bridge the gap between the two communities and favor the integration of graphs within relational OLAP frameworks. However, as discussed, combining graph and traditional cubes is, in the general case, difficult. For this reason, we highlighted the need to keep researching into specific graph warehousing tools that preserve the graph structure and benefit from graph-specific modeling and processing.
Finally, the framework proposed in this paper still requires further physical optimizations, such as custom graph indexing and materialization, that capture

the specific nature of topological and graph-structured cubes (He & Singh, 2006; Zhao, Yu, & Yu, 2007). In our future work, we will focus on the physical optimization using distributed graph processing frameworks such as GraphX.

References	Graph Model	Multidimensional Model				Querying	Scalability
		Dimensions	Measures	Cube	IC		
Graph OLAP (Chen et al., 2009)	Collection of Homogeneous Labeled Graph Snapshots	Info-Dims & Topo-Dims	Traditional, Topological, Aggregate Graph	Graph Cube	-	Roll-up/Drill-down/Slice-dice	Centralized
GraphCube (Zhao et al., 2011)	Homogeneous, Node Attributed	Node Attributes	Aggregate Graph	Graph Cube	-	Cuboid & Crossboid	Centralized
HMGraph (Yin et al., 2012)	Heterogeneous, Attributed	Information, Topological, Entity	Aggregate Graph	HMGraph Cube	-	Rotate, Stretch, Roll-up, Drill-down	Centralized
(Denis, Ghrab, & Skhiri, 2013)	Homogeneous, Node Attributed	Node Attributes	Aggregate Graph	Graph Cube	-	Cuboid & Crossboid	Distributed (Spark)
Pagrol (Wang et al., 2014)	Homogeneous, Attributed	Node and Edge Attributes	Aggregate Graph	Hyper Graph Cube	-	Roll-up & Drill-down	Distributed (Hadoop)
TSMH (Wang et al., 2015)	Heterogeneous, Node Attributed	Node Attribute & Meta-Path	Aggregate Graph	Entity Hyper Cube, Dimension Cube	-	Roll-up, Drill-down, Attribute Transformation	Distributed (Spark)
(Gómez et al., 2017)	Labelled Directed Multi-hypergraphs	Dimension Graphs	Aggregate Graph	Graphoid	-	Climbing and Aggregation, Roll-up, Drill-down, Slice, Dice	Centralized
P&D GraphCube (Wu et al., 2017)	Heterogeneous, Attributed	Node Attribute, Relation Path Set	Aggregate Graph	P&D Graph Cube	-	Entity, Edge, Topological Structure	Distributed (Hadoop, Spark)
(Kang, Lee, & Kim, 2019)	Homogeneous, Attributed	Node and Edge Attributes	Aggregate Graph	Graph Cube	-	Cuboid	Distributed (Spark)
TopoGraph	Property Graph	Content, Topological, Graph-structured	Content, Topological, Graph-structured	Content, Topological, Graph-structured	X	Selection, Roll-up, Drill-down, Drill-across	Centralized

Table 3: Comparison of Graph Cube Frameworks

8 Conclusion and Open Challenges

In this paper, we extended the state of the art on graph warehousing by designing a multidimensional graph model that leverages the content and the topology of the graph. We proposed for the first time a model that exposes both numerical and graph-structured insights using graph cubes, while preserving multidimensional integrity constraints. Furthermore, we proposed different analytical scenarios and formalized the OLAP querying of the graph cubes. We also discussed the potential correspondence between graph cubes and traditional ROLAP cubes. With regard to the implementation, we have detailed the framework architecture and the system API, and evaluated its efficiency with multiple real-world datasets.

As future research direction, our target is to improve the performance of TopoGraph using a distributed graph engine and combine existing graph cube materialization techniques with our novel cube definitions. We also plan to extend our work on dynamic graphs to support continuous update and analysis of real-time graph data. Further work needs to be done on defining a multidimensional query language for graphs, and designing efficient optimization strategies. Machine learning algorithms could also be used to enable advanced mining scenario such discovery of interesting patterns in the graph cube, and the prediction of the graph cube evolution.

References

Akoglu, L., Tong, H., & Koutra, D. (2015). Graph-based Anomaly Detection and Description: A Survey. *Data Mining and Knowledge Discovery*, *29*(3), 626–688.

Benatallah, B., Motahari-Nezhad, H. R., et al. (2015). Scalable graph-based OLAP analytics over process execution data. *Distributed and Parallel Databases*, 1–45.

Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., & Pedreschi, D. (2013). Multidimensional Networks: Foundations of Structural Analysis. *World Wide Web*, *16*(5-6), 567–593.

Chen, C., Yan, X., Zhu, F., Han, J., & Yu, P. S. (2009). Graph OLAP: a multi-dimensional framework for graph data analysis. *Knowl. Inf. Syst.*, *21*(1), 41–63.

Cuzzocrea, A., Bellatreche, L., & Song, I.-Y. (2013). Data Warehousing and OLAP over Big Data: Current Challenges and Future Research Directions. In *Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP* (pp. 67–70). ACM.

Cuzzocrea, A., Saccà, D., & Ullman, J. D. (2013). Big Data: a Research Agenda. In *Proceedings of the 17th International Database Engineering & Applications Symposium* (pp. 198–203). ACM.

Denis, B., Ghrab, A., & Skhiri, S. (2013). A distributed approach for graph-oriented multidimensional analysis. In *2013 IEEE International Conference on Big Data Workshops* (pp. 9–16). IEEE.

Ghrab, A., Romero, O., Jouili, S., & Skhiri, S. (2018). Graph BI & Analytics: Current State and Future Challenges. In *International conference on big data analytics and knowledge discovery* (pp. 3–18). Springer.

Ghrab, A., Romero, O., Skhiri, S., Vaisman, A., & Zimányi, E. (2015). A Framework for Building OLAP Cubes on Graphs. In *East European Conference on Advances in Databases and Information Systems* (pp. 92–105). Springer.

Gómez, L., Kuijpers, B., & Vaisman, A. (2017). Performing olap over graph data: Query language, implementation, and a case study. In *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics* (pp. 1–8). ACM.

He, H., & Singh, A. K. (2006). Closure-Tree: An Index Structure for Graph Queries. In *Proceedings of the 22Nd International Conference on Data Engineering* (pp. 38–). IEEE.

Jin, X., Han, J., Cao, L., Luo, J., Ding, B., & Lin, C. X. (2010). Visual Cube and On-Line Analytical Processing of Images. In *Proceedings of the 19th ACM International Conference on Information and knowledge management* (pp. 849–858). ACM.

Kang, S., Lee, S., & Kim, J. (2019). Distributed Graph Cube Generation using Spark Framework. *The Journal of Supercomputing*, 1–22.

Lenz, H.-J., & Shoshani, A. (1997). Summarizability in OLAP and Statistical Data Bases. In *Proceedings of the Ninth International Conference on Scientific and Statistical Database Management* (pp. 132–143). IEEE.

Leskovec, J., & Krevl, A. (2014, june). *SNAP Datasets: Stanford large network dataset collection.* http://snap.stanford.edu/data.

Li, C., Yu, P. S., Zhao, L., Xie, Y., & Lin, W. (2011). InfoNetOLAPer: Integrating InfoNetWarehouse and InfoNetCube with InfoNetOLAP. *PVLDB*, *4*(12), 1422–1425.

Lin, C. X., Ding, B., Han, J., Zhu, F., & Zhao, B. (2008). Text Cube: Computing IR Measures for Multidimensional Text Database Analysis. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on* (pp. 905–910). IEEE.

Loudcher, S., Jakawat, W., Soriano-Morales, E.-P., & Favre, C. (2015). Combining OLAP and information networks for bibliographic data analysis: a survey. *Scientometrics*, *103*, 471-487.

Petermann, A., Junghanns, M., Müller, R., & Rahm, E. (2014). Graph-based Data Integration and Business Intelligence with BIIIG. *Proc. VLDB Endow.*, *7*(13), 1577–1580.

Qu, Q., Zhu, F., Yan, X., Han, J., Philip, S. Y., & Li, H. (2011). Efficient Topological OLAP on Information Networks. In *Database Systems for Advanced Applications, pages=389–403.* Springer.

Queiroz-Sousa, P. O., & Salgado, A. C. (2019, December). A review on olap technologies applied to information networks. *ACM Trans. Knowl.*

Discov. Data, *14*(1), 8:1–8:25.

Rodriguez, M., & Neubauer, P. (2010). Constructions from Dots and Lines. *Bulletin of the American Society for Information Science and Technology*, *36*(6), 35–41.

Russell, M. A. (2013). *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More.* O'Reilly Media, Inc.

Skhiri, S., & Jouili, S. (2013). Large graph mining: Recent developments, challenges and potential solutions. In M.-A. Aufaure & E. Zimányi (Eds.), *Business intelligence* (Vol. 138, p. 103-124). Springer.

Vaisman, A., & Zimányi, E. (2014). *Data warehouse systems: Design and implementation.* Springer.

van der Aalst, W. M. (2013). Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia-pacific conference on business process management* (pp. 1–22).

Wang, P., Wu, B., & Wang, B. (2015). TSMH Graph Cube: A Novel Framework for Large Scale Multi-dimensional Network Analysis. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (p. 1-10). IEEE.

Wang, Z., Fan, Q., Wang, H., Tan, K.-l., Agrawal, D., & El Abbadi, A. (2014). Pagrol: Parallel graph OLAP over large-scale attributed graphs. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on* (pp. 496–507). IEEE.

Wu, X., Wu, B., & Wang, B. (2017). P&D Graph Cube: Model and Parallel Materialization for Multidimensional Heterogeneous Network. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 95–104). IEEE.

Yin, M., Wu, B., & Zeng, Z. (2012). 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.

Zhao, P., Li, X., Xin, D., & Han, J. (2011). 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.

Zhao, P., Yu, J. X., & Yu, P. S. (2007). Graph Indexing: Tree + Delta <= Graph. In *Proceedings of the 33rd International Conference on Very Large Data Bases* (pp. 938–949). VLDB Endowment.