Graph BI & Analytics: Current State and Future Challenges

Amine Ghrab[1,2], Oscar Romero[3], Salim Jouili[1], and Sabri Skhiri[1]

[1] EURA NOVA R&D, Mont-Saint-Guibert, Belgium
firstname.lastname@euranova.eu
[2] Universitat Politècnica de Catalunya, Spain
oromero@essi.upc.edu

Abstract. In an increasingly competitive market, making well-informed decisions requires the analysis of a wide range of heterogeneous, large and complex data. This paper focuses on the emerging field of graph warehousing. Graphs are widespread structures that yield a great expressive power. They are used for modeling highly complex and interconnected domains, and efficiently solving emerging big data application. This paper presents the current status and open challenges of graph BI and analytics, and motivates the need for new warehousing frameworks aware of the topological nature of graphs. We survey the topics of graph modeling, management, processing and analysis in graph warehouses. Then we conclude by discussing future research directions and positioning them within a unified architecture of a graph BI and analytics framework.

1 Introduction

Graphs are fundamental and widespread structures that provide an intuitive abstraction for the modeling and analysis of complex, heterogeneous and highly interconnected data. They have the benefit of revealing valuable insights from content-based and topological properties of data. The great expressive power of graphs, along with their solid mathematical background, encourages their use for modeling domains having complex structural relationships. In the context of Big Data, the focus of organizations is often on handling the rising volume of their data. However the variety and complexity of data through the different phases of data capturing, modeling and analysis is at least equally important. The variety challenge is the most critical challenge in big data nowadays, and efficiently handling the variety of data sources is considered to be the main driver of success for data-driven organizations [1]. Graphs meet the requirements to be the perfect canonical data model for data integration systems [2] given (1) their capability to deal with semantic relativism and semantic heterogeneities, (2) they are semantically richer at least as any other model (so they can represent any semantics), (3) they allow to create multiple views from the same source, (4) and most importantly, graphs are extremely flexible to compose new graphs. That is given two graphs, with one single edge a new graph could be directly created without affecting the existing ones. Therefore, graphs are suitable to deal with big data variety better than any other data model.

Furthermore, in industry, graph analysis is considered as *"possibly the single most effective competitive differentiator for organizations pursuing data-driven operations and decisions after the design of data capture"* according to Gartner, Inc., a research and advisory firm [3]. Indeed, large complex graphs have emerged in various fields and graph analytics are being increasingly used to solve complex real-world problems. In the financial sector for example, several types of fraud could be detected and prevented in auction and transaction networks [4]. In [5], the authors used bank transaction to build a financial transactions network, where each node represents a client, and each edge represents a transaction. As fraudsters tend to collaborate to orchestrate complex fraud at large scale, the probability that a customer is involved in a fraud depends on his neighborhood in the transaction graph. Graph analytics could be used to define and retrieve complex fraud patterns, or to score customers by fraud exposure. In [6], the authors built a Call Detail Record graph to understand the call routines and interactions between customers. This information can later be used to prepare marketing campaigns or to prevent customer churn.

It is clear that the topological properties of graphs are of big potential to decision-making systems. They supply these systems with a new class of complex structural business facts and measures that could be explored for making more accurate decision in data-driven organizations. In current information systems, Business Intelligence (BI) systems are critical for strategic decision making. Graph BI in particular, is emerging as the BI field that extends current BI systems with graph analytics capabilities. It enables graph-based insights such as detection of popular users or communities in social networks, or revealing hidden interaction patterns in financial networks. Graph BI can help address the above-mentioned big data applications since (1) data is interconnected in complex ways, but graphs can help reduce this complexity with intuitive data models and queries, (2) the data size is large, but data warehouses and Online Analytical Processing (OLAP) analysis are suitable for storage, organization, synthesis and analysis of large volumes of data, and (3) graph mining extends traditional techniques by including discovery of the topological properties, thus characterizing more precisely business applications. Traditional BI 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 not suitable for efficient graph analysis. The structure-driven management and analytics of graph data call for the development of novel data models, query processing paradigms and storage techniques. Therefore, as motivated by multiple research lines [7, 8], current BI and analytics systems need to be extended to efficiently support warehousing [9], processing [10], mining [11] and OLAP analysis [12, 13] of the graph structural and content-based information.

Figure 1 provides an overview of the different components of the envisioned graph BI system. While adopting a similar template as the traditional BI systems (i.e., it preserves the familiar data analytics workflow), graph BI extends current systems with graph-aware components that deliver graph-derived insights.

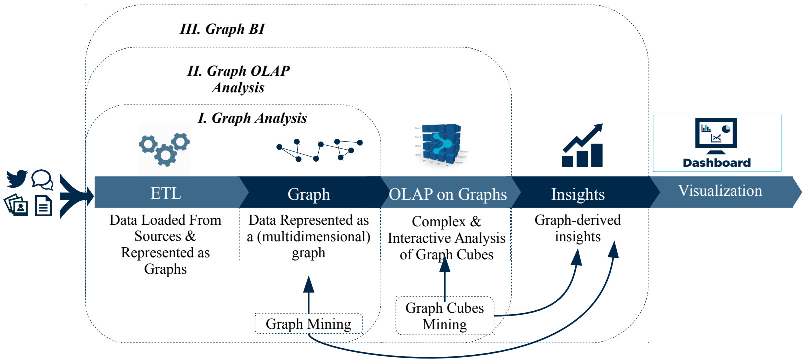

Fig. 1. Integration of Graph and Traditional BI

Note that through this paper the terms "graph and network", "node and vertex", and "edge and relationship" are often used interchangeably. The remainder of this paper presents the current state and the open challenges of graph BI & analytics, with a focus on graph warehousing. Section 2 discusses the topic of graph data modeling and management. Section 3 surveys the existing frameworks for graph analytics. Section 4 identifies future research directions and position them within a unified architecture of a graph BI & analytics framework.

2 Graph Data Modeling

The variety of data structures urges the need for equipping analysts with modeling and querying tools that are aware of the specific nature of each data model. The relational model and its implementations have been developed and matured for decades. However, they were pushed to their limits as the one-size-fits-all data management solutions. The wide adoption of the emerging NoSQL solutions by industry, while not yet as mature as relational model, proved the need to push databases into fields beyond the traditional business applications. More specifically, organizations experience an urgent need for models and techniques for efficient management of graph data. Indeed, graph models have the ability to deal with semantic relativism and heterogeneities, offer the flexibility to combine graphs, and support the capability to associate data and metadata.

2.1 Graph Models

According to the literature, the two main families of graphs are property graphs and knowledge graphs:

Property Graphs : Property graphs describe a directed, labeled and attributed multi-graph. Each real-world entity is represented by a node that contains its label and properties. The label denotes the "type" of the node (i.e., the class to which it belongs). Relationships between entities are represented using edges. The flexibility of property graph models allows the representation of rich structural properties, such as hierarchies and assertions. Property graphs were introduced in the database community to store schemaless data (due to their flexibility to absorb any semantics and attach data with metadata). In the literature, multiple query languages were designed to enable graph-oriented querying of property graphs [14]. However, there is no standard query language for property graphs. Therefore, graph database vendors defined their own graph traversal and query languages, such as Cypher and Gremlin. Cypher is an SQL-like declarative language, that uses isomorphism-based no-repeated-edges bag semantics. It was introduced by Neo4j, and is centered around pattern matching enriched by built-in algorithmic operators. Gremlin is a graph traversal language, built using Groovy, introduced by Apache TinkerPop3, that uses the homomorphism-based bag semantics.

Knowledge Graphs: The basic formalism behind knowledge graphs, used to describe and link resources, is the Resource Description Framework language (RDF), a W3C recommendation. The basic RDF block is the triple, a binary relationship between a subject and an object; i.e., $< subject, predicate, object >$. The subject and the predicate must be resources (i.e., identified by a URI), whereas the object can be either a resource or a literal (i.e., a constant value such a string or an integer). A set of RDF triples form an RDF graph. RDF Schema (RDFS), a W3C recommendation, was introduced to express basic constraints on RDF triples. In the same line, the Ontology Web Language (OWL) allows to express richer constraints and semantics. As OWL is serialized on top of RDFS, it results in a graph too. Importantly, it is also usual to refer to knowledge graphs (i.e., RDF(S) or OWL) as ontologies. This is because they can be translated to a fragment of First Order Logic (FOL), typically, within the family of Description Logics (e.g., in the case of OWL and OWL fragments), and benefit from generic reasoning algorithms in that field. Finally, the W3C recommendation to query knowledge graphs is the SPARQL Protocol and RDF Query Language (SPARQL). Relevantly, SPARQL enables to activate generic reasoning capabilities when querying knowledge graphs to infer non-explicit knowledge. Knowledge graphs were born within the semantic web stack and therefore initially thought for enabling interoperability and reasoning on semantic data. They are tightly related to the knowledge representation community and therefore thought to represent generic knowledge rather than data as for databases. For this reason, knowledge graph databases are referred to as triplestores rather than graph databases. One key aspect with regard to property graphs is that they provide means (i.e., URI) to universally identify graph vertices and edges from external sources. This facilitates linking and sharing of data and metadata. However, unlike property graphs and traditional graph databases, which are optimized for graph traversal, they are primarily optimized for handling RDF triples. Another difference is that in property graphs properties could be directly added to edges as well as vertices. In essence, however, knowledge graphs are also graphs and can benefit from the traditional graph algebras presented in the database field.

2.2 Graph Management

Orthogonal to the previous classification there are two main approaches widely used for graph data management (regardless of property or knowledge graphs) at the logical/physical level. The first consists on the use of native graph data models and database engines. The second leverages alternative models, mainly the relational model. For the latter, the graph data is represented by a set of tables, i.e., node tables and edge tables. Traditionally, relational-based graph database engines have been related to triplestores and knowledge graphs, whereas native graph database engines were related to property graphs. Nevertheless, this is nowadays changing and it is currently possible to find native databases for knowledge graphs. These approaches are discussed next:

Native Graph Data Models : In recent years, the trend in developing graph data management systems has shifted to the development of native, relationship-oriented graph databases. Most native graph databases implement a the property graph model or a variation of it. The problem of impedance mismatch is resolved since relationships are first-class citizen, and the data is represented as it is perceived without the need to project it on an intermediate representation. The data model is more straightforward to design, and the queries are more intuitive to formulate [15]. From a performance perspective, graph databases are optimized for graph traversals. The cost of traversing an edge is constant, and the overall cost of arbitrary navigation through the graph is much lower than the equivalent joins in relational tables. Subsequent implementation aspects such as graph query processing, indexing, storage, matching and retrieval which are specifically developed and tuned for graph workloads lead to better performances, especially for queries requiring multiple joins, or containing cycles or other complex patterns. However, they perform worse than the relational-based engines for analytical queries that perform scans over the whole graph. In the software market, established BI vendors are aware of the potential of native graph solutions and have already developed many graph databases such as Neo4j, DataStax Enterprise Graph, Oracle Spatial and Graph, Microsoft GraphEngine, IBM Graph, and Amazon Neptune.

Relational-based Database Models : This approach benefits from the well-established relational model features, and enables a smooth integration with a wide range of relational platforms. However, the relational model and its implementations fall short of meeting the requirements for (1) intuitive data modeling, (2) topology-aware graph querying (such as path retrieval and comparison, and graph pattern matching), and (3) traversal-optimized performances. Mapping graph data to relational representation raises the problem of impedance mismatch at the modeling and querying levels. For example, due to the fundamental difference between the two models, transformation of graph data to the relational model is a manual, complicated process, with a high risk of information loss during the transformation process. The relational model was designed to handle data such as sets of records and transactions instead of entities and connections. The relational query languages and query processing engines are optimized to perform table scans instead of traversals. Graph traversal is often simulated using expensive join operations, which incurs a heavy workload especially for highly interconnected tables. Moreover, the SQL is not suited to target the topology of the graph with queries such as pattern matching, neighborhood or path retrieval.

3 Graph Analytics

A plethora of graph analysis techniques were proposed in the literature to reveal interesting properties about the graph topology and the connectivity between graph elements. The core analysis operations of graphs are (1) graph traversal to assess reachability, find shortest paths, and retrieve the neighborhood, (2) metrics computation of local (e.g., centrality), and global properties (e.g., diameter), and (3) graph matching, such as subgraph isomorphism and pattern matching [14]. Most of these operations are supported by graph database engines. This section describes more advanced graph analytics and focuses on three pillars of graph analytics: graph OLAP, graph mining, and graph processing.

3.1 OLAP on Graphs

The multidimensional model is widely used to represent data in the warehouse. The business facts are stored following the multidimensional model in cubes that embed aggregated data denoted as measures, which are the metrics for the analysis. The measures are placed into the so-called multidimensional space, where dimensions are the factors influencing the values of the measures. OLAP techniques are widely used by BI analysts to conduct interactive and complex querying over large volume of data, from different perspectives and through different hierarchical levels, highlighting the items of interest, and then drilling down to the underlying data from which facts were inferred. The main approaches for the multidimensional design and OLAP analysis of cubes on graphs are:

Graph OLAP was among the first attempts to design a conceptual framework for OLAP analysis over a collection of homogeneous graphs [12]. Each graph of the collection is considered as a snapshot. Attributes are considered as the dimensions, and could be either attached to the whole graph snapshot, or to single nodes. In the first case, the attributes of the snapshots are called informational dimensions. The aggregations of the graph are performed by overlaying a collection of graph snapshots and merging those with shared informational values. The analysis is referred to as informational OLAP aggregations, and consists in edge-centric snapshot overlaying. Thus only edges are merged and changed, with no changes made to the nodes. In the second case, the attributes of the nodes are called topological dimensions. Topological OLAP aggregations consist on merging nodes and edges by navigating through the nodes hierarchy. Qu et al. introduced a more detailed framework for topological OLAP analysis of graphs [16]. The authors discussed the structural aggregation of the graph following the OLAP paradigm. They presented techniques based on the properties of the graph measures for optimizing measure computations through the different aggregation levels. Berlingerio et al. [17] defined a multidimensional model similar to Graph OLAP, but where the dimensions are the labels of the edges, and presented a set of analytical graph-based measures.

Graph Cube is a framework for multidimensional analysis and cube computation over the different levels of aggregations of a graph [18]. It targets single, homogeneous, node-attributed graphs. A subset of the attributes of the nodes is considered as the dimensions. Following these so-called dimensions, the graph cube is obtained by restructuring the initial graph in all possible aggregation. Given n dimensional attributes, the framework introduces the cuboid query, which generates 2^n aggregate graphs (called graph cuboids). Crossboid is a second query introduced by Graph Cube to analyze the interrelationships between different graph cuboids. Pagrol is a Map-Reduce framework for distributed OLAP analysis of homogeneous attributed graphs [19]. Pagrol introduced the notion of Hyper Graph Cubes that extends the model of Graph Cube by in addition considering the attributes of the edges as dimensions, and introduced various optimization techniques for cubes computation and materialization. Ghrab et al. [20] extended those models with a framework for building OLAP cubes on heterogeneous attributed graphs. They presented an extension of property graphs tailored for multidimensional analysis and supporting dimension hierarchies.

Graph OLAP on RDF There is an active research line to generate OLAP cubes on top of RDF and RDF(S) graphs. Nebot [21] and Kämpgen [22] were two of the main attempts to bridge both areas. The former proposes a semi-automatic method for on-demand extraction of semantic data into an MD database. In this way, data could be analyzed using traditional OLAP techniques. The latter study the extraction of statistical data published using the QB vocabulary, a W3C standard, into an MD database. Both approaches moved the semantic data to a traditional data warehouse. Subsequent attempts avoided such approach and query graph data in an OLAP manner without moving it. For example, Beheshti et al., introduced a distributed framework for OLAP on RDF data [23]. 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. However, it was not until the definition of the QB4OLAP vocabulary that cubes on RDF graphs could not be guaranteed to be MD-compliant. In [24], Varga et al. discusses the drawbacks of previous vocabularies, such as QB, to properly represent MD data and how QB4OLAP overcomes them. This way, resulting cubes can be properly analyzed with traditional OLAP algebras. Relevantly, Pratap Deb Nath et al.

present a framework to conduct ETL transformations on top of graph data to produce QB4OLAP-based cubes [25].

3.2 Graph Mining

Data mining refers to the process of discovering patterns or models for data. In contrast to querying that retrieves known patterns, mining enable the discovery of previously unknown, implicit information and knowledge embedded within a dataset. Traditionally, data mining techniques process data as collection of independent instances (i.e. observations). However, the recent emergence of graph structure as a rich data model involves a paradigm shift on how data mining algorithms can be applied. In fact, graph mining algorithms provide a new way of extracting and discovering latent insight from graphs by leveraging the relationships between entities. However, graph mining algorithms face three main challenges: (1) adapting the mining algorithms to make them graph-aware, (2) redesigning the algorithms to be implemented by those new high-performance techniques, and (3) storing and exploiting multiple but related graphs that serve for the same business purpose as in the graph warehouse.
A plethora of graph mining techniques were proposed in the literature such as graph clustering, frequent subgraph mining, proximity pattern mining and link prediction. These techniques are relevant in the BI context as they reveal interesting properties about the topology and the connectivity between business entities. For example, consider the case of recommender systems in e-commerce [26]. The domain could be a represented as a bipartite purchase graph, with two types of nodes representing products and customers. An edge is added between a product and a customer if the latter has bought the product. Using graph mining such as graph clustering, two other graphs could be derived: (1) client similarity graph, and (2) product similarity graph. Mining these three graphs enables advanced analysis scenarios such as (1) customer profiling by detecting customer groups using community detection, and the leaders within each group using centrality, (2) product segmentation by detecting products representative of each segment, and (3) using link prediction, targeted marketing personalized to the customer's profile and tailored by current product trends.

The historical and integrated view provided by the data warehouse makes it a suitable backbone for offering a variety of analysis scenarios. In the graph warehouse context, graph mining could be combined with OLAP to offer more capabilities both during the phase of the design and also the analysis of the graph warehouse data. During the design phase, graph mining algorithms could be used to enrich the OLAP cubes with new types of topological dimensions and measures (e.g., PageRank, community). During the analysis, graph mining could assist the analyst in complex tasks such as building summarized business-oriented views of the graph, providing new perspectives to analyze the graph, or discovering interesting or anomalous patterns within the large graph cube space. In the context of outlier detection, graphs provide an elegant framework to predict and describe outliers. For example, in the context of graph cubes mining, [27] developed a measure of interestingness of patterns in a graph cube, while [28] proposed an entropy-based filter to detect interesting associations between attributed nodes in a graph cube.

3.3 Graph Processing

To deal with large and evolving graphs, which is the case in data warehouses, graph BI systems need to integrate large-scale graph processing frameworks. Graph processing frameworks are designed to natively support the graph topology, and they offer graph programming models and abstractions to easily implement a multitude of graph algorithms. These frameworks have the capabilities to efficiently perform large scale, ad-hoc, and distributed computations over large graph data that exceed a single machine capacity. They offer features such as automatic graph partitioning, load balancing, network and disk transfer optimization, and fail-over of the processing tasks.

However, distributed graph processing poses additional challenges to centralized or traditional parallel data processing in that [29]: (1) graph structure is irregular which poses challenges to the graph data partitioning and limits parallelism, (2) computation is driven by the structure, which causes a poor memory locality and poses data transfer issues, and (3) algorithms traverse the partitioned graph in an exploratory way, and are iterative by nature, which is I/O intensive. To tackle these challenges and enable efficient large-scale graph analytics, different processing paradigms were introduced [30]:

- Hadoop Family frameworks: MapReduce denotes a programming model for large-scale data processing. Hadoop is an open-source framework that supports data-intensive distributed applications and clones the Google's MapReduce framework. It is designed to process large amount of unstructured and complex data and runs on shared-nothing architectures. MapReduce frameworks are useful for content-based aggregation of graphs (e.g., graph cube aggregation), but they are not efficient for graph-specific computations [31].
- Synchronous frameworks: Pregel [32], and its open source implementation Apache Giraph, are distributed fault-tolerant graph processing frameworks designed to execute vertex-centric graph algorithms following the Bulk Synchronous Parallel processing (BSP) paradigm. BSP is a shared-nothing processing paradigm for parallel algorithms execution. The computation is done as a series of super-steps over a set of processing units, each having its local memory. Each super-step consists of three phases, first (1) each processing unit performs concurrently and locally its computations, then (2) data is exchanged between the different processes, finally (3) when a process finishes the computation and communication, it reaches the synchronization barrier and it waits for the rest of processes to finish before proceeding to the next super-step. The advantage of this paradigm is that it ensures a deadlock-free computation. However, the downside is the execution time, where the system has to wait for the slowest machines to finish before proceeding.
- Asynchronous frameworks: In contrast to the synchronous shared-nothing processing frameworks, GraphLab [33] and PowerGraph [34] are asynchronous

and follows the the Gather-Apply-Scatter computational model, with shared memory abstraction. These frameworks might provide better performances, but incur more complexity and higher scheduling and consistency costs.

- Hybrid Systems: These frameworks enable a mixed workload of graph-parallel and data-parallel processing. GraphX [35] is a component of Apache Spark [36] developed for graph processing . It is a fault-tolerant, distributed, in-memory graph processing framework built on top of the Resilient Distributed Dataset abstraction. GraphX provides a set of primitive operators to load and interactively query the graph data. GRADOOP is a distributed framework for graph management and querying [37]. It introduces a new graph model that extends property graphs, support Cypher queries, and the queries are processed using of Apache Flink [38].

4 Future Research Directions

This paper calls for the development of novel intelligent, efficient and industry-grade graph warehousing systems. The potential directions include (1) further research on solving complex graph problems (e.g., subgraph isomorphism and graph partitioning), (2) building native graph components (e.g., native graph ETL operations, graph OLAP engines, and a multidimensional query language for graphs), and (3) the integration of artificial intelligence techniques to enable self-service BI for business users. To this end, machine learning should be integrated within the BI systems to automate the warehousing tasks from data preparation, to complex modeling and augmented analytics of graphs (e.g., automated discovery of multidimensional concepts, detection of interesting patterns, and forecasting of business KPIs evolution).

The modules missing for developing an industry-grade graph BI and analytics system are unified in the envisioned architecture presented in Figure 2, and they summarize the future research directions as follows:

- Graph Extraction (1): This module allows the extraction of graph data from different data sources that could initially be in various formats and flowing at various rates such as graph streams. The data is cleaned to only capture entities that satisfy the quality constraints (e.g., contains the required attributes with valid values), which guarantees the reliability of data.
- Graph Construction & Enrichment (2): The captured graph data is integrated and formatted according to a given graph model. A promising research direction is the development of graph-aware ETL processes with native graph manipulation operations augmented with machine learning capabilities. ML techniques such as Information Retrieval and Automatic Natural Language Processing could help in the automated extraction and construction of multidimensional graphs from unstructured data such as text. For example, geolocation and sentiment analysis could be applied to enrich the attributes of the data entities and therefore equip businesses with the capability to analyze data from new perspectives. New graph entities could be discovered as well. For instance, using community detection, new labels could be added

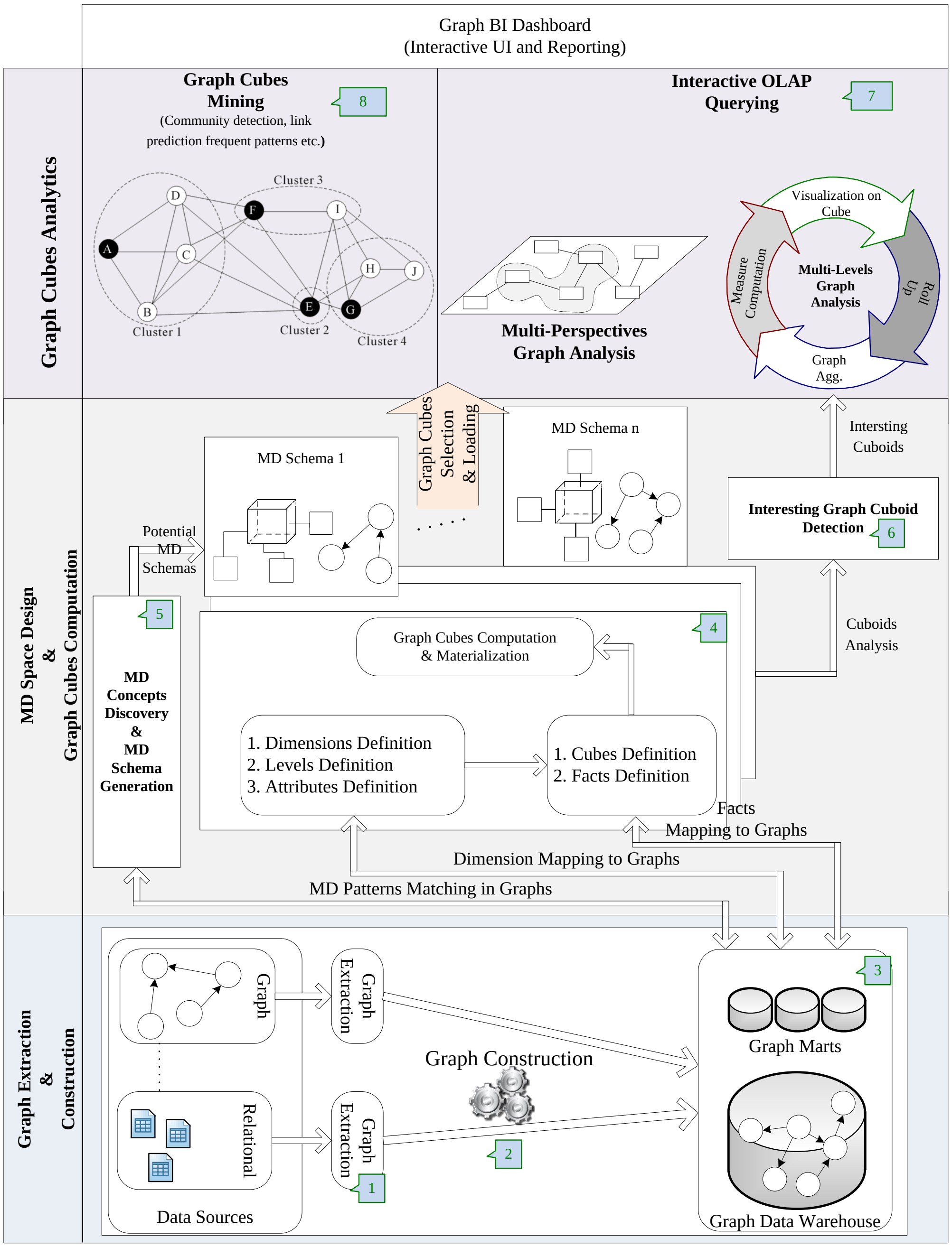

Fig. 2. Architecture of a Graph BI & analytics framework

to the nodes, and using similarity, new edges could be added between the similar node. Multiple variation of the traditional ETL approach exist in the literature, and might be worth exploring for Graph ETL such as the Extract-Load-Transform, or the Capture-Transform-Flow.

- Graph Data Warehouse (3): The graph data warehouse is the reference central information repository for graph-based decision making. Data is extracted from different sources and integrated using a common graph model. The cleansed and integrated data is natively stored and managed as a multidimensional graph in the graph warehouse. Whereas that data would be transformed to tables in traditional data warehouses. Nevertheless, the conceptual layer remains the same (i.e., represented as dimensions and facts). The changes are related to the logical and physical levels. The graph warehouse provides a suitable backbone for natively analyzing graphs with BI tools such as graph OLAP and graph mining.
- Cube Design and Computation (4): The semantic relativism inherent in graphs allows to create several views from the same data and make them co-exist in a much simpler way than other data models. Afterwards, given a graph lattice, the graph cubes enable the computation and the aggregation of corresponding graph cuboids. Once the required graph cuboids are computed, the result is persisted in the corresponding data marts. To leverage graph properties, graph cubes embed graph-structured measures and dimensions. There is a need for cube computation and aggregation libraries capable of efficiently handling graphs. This line of research includes optimizations such as graph cuboids materialization, indexing, and graph icebergs.
- Discovery of multidimensional concepts and definition of potential multidimensional schemas (5): Multiple multidimensional schemas could be built from the same graph warehouse to satisfy the various analysis needs. Real-world graphs, such as social networks, are complex, dynamic and flexible. Interesting graph entities might be hidden in the large data sources. Therefore, there is need for novel graph-aware approaches that enables automatic detection and extraction of multidimensional concepts from large complex graphs. This could be done through the analysis of the topological aspects of graphs, and the projection of the properties of the multidimensional models on them. This will help end-users cope with the complexity and large volume of graphs, and expose potential interesting discovery to decision makers.
- Assistance with the analysis and synthesis of graphs (6): Given the complexity and large size of the initial graph, there is need for intelligent modules capable of performing automated preliminary analysis of the graph to guide the analyst during the exploration of the graph cubes. The goal is to enable self-service BI and facilitate complex tasks such as the extraction of meaningful graph summaries, the discovery of interesting phenomena in the graph cuboids such as frequent graph patterns and outlier relationships.
- Mining and querying OLAP cubes (7-8): Complex and interactive OLAP analysis and mining of graph cubes is performed at this phase. In contrast to traditional OLAP, graph cubes enable the multidimensional analysis of graph metrics stored in the graph cuboids. For example, analysts could examine the

centrality of leaders from multiple perspectives, or identify the communities and their connections at different levels of aggregations. To this end, there is a need to develop graph OLAP engines that support graph-structured cubes. In addition, Online Analytical Mining of graph data is a promising research direction to empower graph OLAP with mining capabilities. Graphs are dynamic and enabling OLAP on evolving networks by analyzing changing facts and dimensions will help in understanding the structural and informational evolution of networks. Many BI vendors have already integrated graphs into their BI solutions. However, the support for graphs is still limited and there is still a need to push further the integration of graph-derived insights into the decision-making process.

5 Conclusion

Graphs are interesting structures that provide a solid foundation for intuitively representing various domains and solving complex problems, while enabling better performance. Graph analytics leverage structural and content-based information to create added-value services, and extend current solutions with new topology-enabled capabilities. In this paper, we surveyed the state of the art on graph BI and analytics, and proposed an architecture of Graph BI and Analytics platform augmented with machine learning capabilities, which lays the foundations for promising future research directions. In all, graph analytics have a bright future, and this paper calls for more attention from academia and industry to build next-generation graph-powered BI and analytics frameworks.

References

1. Bean, R.: Variety, Not Volume, Is Driving Big Data Initiatives. https://sloanreview.mit.edu/article/variety-not-volume-is-driving-big-data-initiatives/ (2016) Accessed: 2018-01-25.
2. García-Solaco, M., Saltor, F., Castellanos, M. In Bukhres, O.A., Elmagarmid, A.K., eds.: Object-oriented Multidatabase Systems. Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK (1995) 129–202
3. Feinberg, D., Heudecker, N.: IT Market Clock for Database Management Systems. https://www.gartner.com/doc/2852717/it-market-clock-database-management (2014) Accessed: 2018-01-02.
4. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Mining and Knowledge Discovery **29**(3) (2015) 626–688
5. Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B.: Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. Decision Support Systems **75** (2015) 38–48
6. Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A.A., Joshi, A.: Social ties and their relevance to churn in mobile telecom networks. In: Proceedings of the 11th international conference on Extending database

technology: Advances in database technology. EDBT '08, New York, NY, USA, ACM (2008) 668–677

7. Duan, L., Da Xu, L.: Business intelligence for enterprise systems: A survey. IEEE Transactions on Industrial Informatics **8**(3) (2012) 679–687
8. Lim, E.P., Chen, H., Chen, G.: Business intelligence and analytics: Research directions. ACM Transactions on Management Information Systems (TMIS) **3**(4) (2013) 17
9. Cuzzocrea, A., Bellatreche, L., Song, I.Y.: 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, ACM (2013) 67–70
10. Skhiri, S., Jouili, S.: Large graph mining: Recent developments, challenges and potential solutions. In Aufaure, M.A., Zimányi, E., eds.: Business Intelligence. Volume 138 of Lecture Notes in Business Information Processing. Springer Berlin Heidelberg (2013) 103–124
11. Shi, C., Li, Y., Zhang, J., Sun, Y., Philip, S.Y.: A survey of heterogeneous information network analysis. IEEE Transactions on Knowledge and Data Engineering **29**(1) (2017) 17–37
12. 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) (2009) 41–63
13. Hannachi, L., Benblidia, N., Boussaid, O., Bentayeb, F.: Community cube: a semantic framework for analysing social network data. International Journal of Metadata, Semantics and Ontologies **10**(3) (2015) 155–169
14. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoč, D.: Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR) **50**(5) (2017) 68
15. Hölsch, J., Schmidt, T., Grossniklaus, M.: On the performance of analytical and pattern matching graph queries in neo4j and a relational database. In Ioannidis, Y.E., Stoyanovich, J., Orsi, G., eds.: Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference (EDBT/ICDT 2017), Venice, Italy, March 21-24, 2017. Volume 1810 of CEUR Workshop Proceedings., CEUR-WS.org (2017)
16. Qu, Q., Zhu, F., Yan, X., Han, J., Philip, S.Y., Li, H.: Efficient topological OLAP on information networks. In: Database Systems for Advanced Applications, Springer (2011) 389–403
17. Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., Pedreschi, D.: Multidimensional networks: foundations of structural analysis. World Wide Web **16**(5-6) (2013) 567–593
18. 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, ACM (2011) 853–864
19. Wang, Z., Fan, Q., Wang, H., Tan, K.l., Agrawal, D., El Abbadi, A.: Pagrol: PArallel GRaph OLap over Large-scale Attributed Graphs. In: Data Engineering (ICDE), 2014 IEEE 30th International Conference on, IEEE (2014) 496–507
20. Ghrab, A., Romero, O., Skhiri, S., Vaisman, A., Zimányi, E.: A framework for building olap cubes on graphs. In: East European Conference on Advances in Databases and Information Systems, Springer (2015) 92–105
21. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. Decision Support Systems **52**(4) (2012) 853–868
22. Kämpgen, B., Harth, A.: Transforming statistical linked data for use in olap systems. In: Proceedings of the 7th international conference on Semantic systems, ACM (2011) 33–40

23. Beheshti, S.M.R., Benatallah, B., Motahari-Nezhad, H.R.: Scalable graph-based olap analytics over process execution data. Distributed and Parallel Databases **34**(3) (Sep 2016) 379–423
24. Varga, J., Vaisman, A.A., Romero, O., Etcheverry, L., Pedersen, T.B., Thomsen, C.: Dimensional enrichment of statistical linked open data. Web Semantics: Science, Services and Agents on the World Wide Web **40** (2016) 22–51
25. Nath, R.P.D., Hose, K., Pedersen, T.B., Romero, O.: Setl: A programmable semantic extract-transform-load framework for semantic data warehouses. Information Systems (2017)
26. Lee, K., Lee, K.: Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items. Expert Systems with Applications **42**(10) (2015) 4851–4858
27. Demesmaeker, F., Ghrab, A., Nijssen, S., Skhiri, S.: Discovering interesting patterns in large graph cubes. 2017 IEEE International Conference on Big Data (Big Data) (2017) 3322–3331
28. Bleco, D., Kotidis, Y.: Entropy-based selection of graph cuboids. In: Proceedings of the Fifth International Workshop on Graph Data-management Experiences & Systems, ACM (2017) 2
29. Lumsdaine, A., Gregor, D., Hendrickson, B., Berry, J.: Challenges in parallel graph processing. Parallel Processing Letters **17**(01) (2007) 5–20
30. Batarfi, O., El Shawi, R., Fayoumi, A.G., Nouri, R., Barnawi, A., Sakr, S., et al.: Large scale graph processing systems: survey and an experimental evaluation. Cluster Computing **18**(3) (2015) 1189–1213
31. Denis, B., Ghrab, A., Skhiri, S.: A distributed approach for graph-oriented multidimensional analysis. In: Big Data, 2013 IEEE International Conference on. (Oct 2013) 9–16
32. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, ACM (2010) 135–146
33. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed graphlab: A framework for machine learning and data mining in the cloud. Proc. VLDB Endow. **5**(8) (April 2012) 716–727
34. Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., Guestrin, C.: Powergraph: Distributed graph-parallel computation on natural graphs. In: OSDI. Volume 12. (2012) 2
35. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: Graph processing in a distributed dataflow framework. In: OSDI. Volume 14. (2014) 599–613
36. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing. HotCloud'10, Berkeley, CA, USA, USENIX Association (2010) 10–10
37. Junghanns, M., Petermann, A., Gómez, K., Rahm, E.: Gradoop: Scalable graph data management and analytics with hadoop. arXiv preprint arXiv:1506.00548 (2015)
38. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **36**(4) (2015)