SynopSys: Foundations for Multidimensional Graph Analytics

Michael Rudolf[1], Hannes Voigt[1], Christof Bornhövd[2], and Wolfgang Lehner[1]

[1] TU Dresden, Database Technology Group, Germany
michael.rudolf01@sap.com, hannes.voigt@tu-dresden.de,
wolfgang.lehner@tu-dresden.de
[2] SAP Labs LLC, Palo Alto, CA 94304, USA
christof.bornhoevd@sap.com

Abstract. The past few years have seen a tremendous increase in often irregularly structured data that can be represented most naturally and efficiently in the form of graphs. Making sense of incessantly growing graphs is not only a key requirement in applications like social media analysis or fraud detection but also a necessity in many traditional enterprise scenarios. Thus, a flexible approach for multidimensional analysis of graph data is needed. Whereas many existing technologies require up-front modelling of analytical scenarios and are difficult to adapt to changes, our approach allows for ad-hoc analytical queries of graph data. Extending our previous work on graph summarization, in this position paper we lay the foundation for large graph analytics to enable business intelligence on graph-structured data.

Key words: Graph Databases, Graph Analytics, Graph OLAP

1 Introduction

In the past decade graph data has become abundant. In the form of social networks and road networks it pervades everyday life, but the capability of storing and processing graph-structured data has also become crucial in enterprise tasks, such as supply chain management and product batch traceability. As the amount of graph data grows at an ever-increasing pace, the need for flexible graph analysis technology becomes more and more important. Whereas data warehousing tools and online analytical processing (OLAP) solutions for relational data are well-understood and mature, approaches for multidimensional analysis of graph data are still in their infancy (cf. Section 2). In the context of our research project SynopSys we aim at closing this gap.

There is a plethora of graph models used in research, each tailored to the specific problem at hand. In its most basic definition a graph $G \coloneqq (V, E)$ is a tuple consisting of a set of vertices V and a relation $E \subseteq V \times V$ denoting the edges between them. For the remainder of this paper we employ the property graph model [1] where vertices and edges can have attributes and edges are directed links between pairs of vertices. Its generality allows other graph models to be

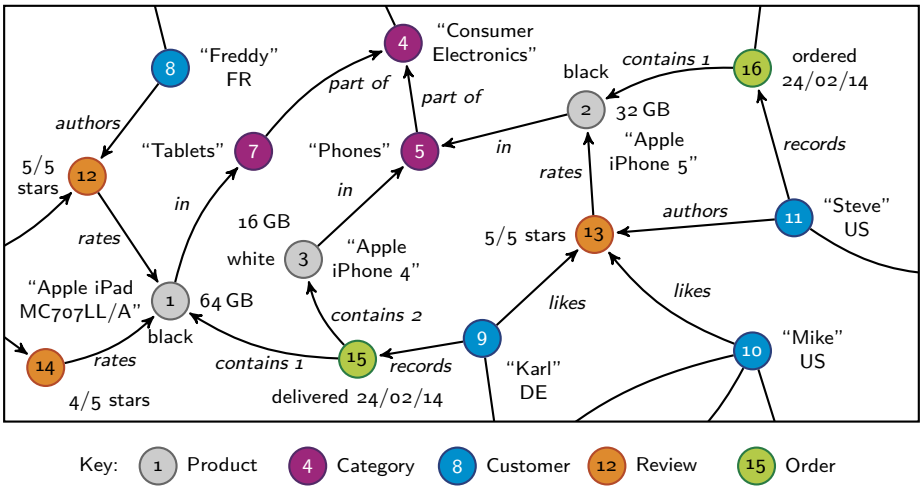

Fig. 1. Property graph illustrating a typical business scenario.

mapped to it easily while being flexible enough to support a broad variety of use cases. Figure 1 shows an excerpt of a property graph capturing a typical business scenario consisting of products, categories, customers, orders, and customer reviews. Note that vertices and edges of the same "type" can differ with regards to the attributes they expose. This scenario will serve as a running example throughout the following sections.

In our previous work [2] we have already outlined the use of graph pattern matching and transformation for deriving graph summaries. By means of summarization rules instantiated from an assorted set of templates, graph data can be grouped and aggregated numbers computed.

In this paper we extend our approach for graph analytics with the well-known concepts from multidimensional analytics [3]. We introduce a formal description of a general-purpose data model linking the concepts *fact*, *dimension*, and *measure* to the property graph model (cf. Section 3). Also, we present the semantics of operations for creating graph summaries along dimensions (cf. Section 4). Finally, we outline our plans for future research (cf. Section 5).

2 Related Work

The multidimensional model [3] is well-established as the theoretical foundation of the vast majority of online analytical processing (OLAP) tools. Data warehouses are usually created by designing a schema containing facts and dimensions. In an analytical session a set of measures has to be defined, and a (hyper-) cube is then constructed to capture both measures and dimensions. Unfortunately this intensional approach of up-front modelling is unable to meet the requirements of today's ever-changing IT and business landscapes with the dramatic increase

of data volumes to process and number of sources to integrate. For a specific analytical session users should be able to specify in an ad-hoc manner what the facts and dimensions of interest are.

In 2008 Chen et al. presented their approach for extending OLAP to graphs [4]. They introduce two kinds of dimensions and map the well-known OLAP operations roll-up, drill-down, and slice/dice to them. *Informational dimensions* are derived from the attributes associated with snapshots of an evolving graph, whereas *topological dimensions* stem from the attributes of vertices and edges in each such snapshot. Building on that, they describe a theoretical foundation for aggregated graphs, which form the measures in OLAP terminology, and propose the use of partial materialization techniques for reducing memory consumption. However, their approach is not accompanied by any processing or evaluation specification, concepts, or architecture. Also, singularizing the temporal dimension of system time does not seem to be sufficient, because application time is often more relevant in practice, and it unnecessarily complicates the formal framework.

In the same year Tian et al. proposed an operator for summarization by grouping nodes on attributes and pairwise relationships (SNAP) [5]. Whenever a vertex of one group is connected to any vertex of the other, the two vertex groups will also be connected. In practice this behavior turns out to be quite limiting, because it can result in a large number of groups. Therefore, they propose the *k-SNAP* operation as an extension, where the homogeneity constraint for group relationships is relaxed and the user can specify the number of groups in the graph summaries. By changing the parameter k, the user can emulate the OLAP operations drill-down and roll-up. The authors prove that the computation of the *k-SNAP* operation is NP-complete and propose heuristics to approximate it. Although the two proposed operations are designed to work with different edge types, additional edge attributes are not supported. Furthermore, there is no support for an independent filtering of the input graph (similar to slicing or dicing in OLAP terms).

In a follow-up paper from 2010 Zhang et al. [6] improve the previous approach in two ways: first, they provide an automatic means of discretizing numerical attributes based on the user-specified number of partitions and the graph topology. Second, based on what they call the diversity, coverage, and conciseness, the authors define an *interestingness* measure for graph summaries. This is then used for helping users to specify a sensible number of groups for the *k-SNAP* operation. The paper improves the practical usability of the overall approach, but does not address the limitations discussed above.

In 2011 Zhao et al. introduced a novel data warehousing model called Graph Cube [7]. Their notion of a multidimensional network is based on a restricted graph model (e.g., no attributes on edges) with the dimensions being the vertex attributes. An aggregate network (called *cuboid*) is then formed by computing equivalence classes for vertices according to the chosen dimensions and by constructing a weighted graph. All possible aggregations of the original network then form a *graph cube*. The authors propose two kinds of OLAP operations: cuboid and crossboid queries. The former simply returns the aggregate network of the

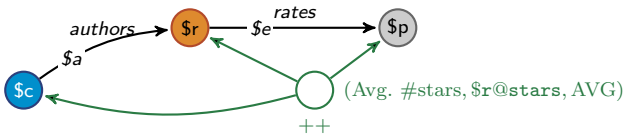

Fig. 2. Summarization rule for selecting customers and their review as facts and for computing a measure for the average number of stars of customer reviews.

desired cuboid from the graph cube, while the latter is somewhat similar to a join operation between multiple different cuboids. As for the specification and evaluation of such queries, the authors do not propose any mechanism but focus on partial materialization techniques instead.

There are various other approaches to summarizing graphs, but most of them are statistical in nature—computing a number of figures (e.g., degree distributions, hop-plots, and clustering coefficients), which describe some characteristics of the graph. The approaches presented above are different in that they can produce aggregated views on the graph data in various, user-controlled resolutions by means of OLAP-like operations. Although much more flexible, in our opinion these approaches are still too rigid, because they fix facts and dimensions up-front.

3 Data Model

In this section we introduce the various elements of the data model underlying our approach for graph analytics. It is heavily inspired by the well-known multidimensional model for analytics in data warehouses [3] but is crafted as a separate layer on top of the property graph model [1].

3.1 Facts

The most fine-grained elements of interest are called *(base) facts*. In the context of graph data, a fact can be an attribute of a vertex or an edge or the presence of an edge itself. For example, interesting facts from our running example can be the price of products, the amount of products in an order, or the existence of an edge between products and reviews.

We propose to specify facts using *summarization rules*, i.e. additive graph transformations rules consisting of a graph pattern and an action for creating a representative. Figure 2 depicts such a summarization rule for selecting customers and the reviews they have authored as facts. Each pattern element is assigned an alias (e.g., "$c" for customer vertices), which can be used for specifying dimensions and measures. The action part of the transformation is colored green and annotated with "++" to indicate the addition of a so-called *representative vertex*. Note that these vertices need not necessarily be materialized; their primary purpose is to provide references to all facts that are relevant for the analytical scenario at hand. The use of graph transformations enables the specification of

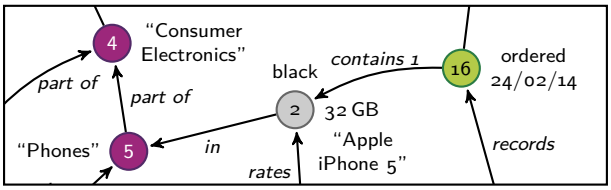

Fig. 3. Explicit and implicit dimensions in a property graph.

complex patterns consisting of both informational and topological predicates in a graphical form, and it does not require the user to learn a dedicated domain-specific language.

3.2 Dimensions

As the name suggests, the principal element of multidimensional analysis is the concept of a *dimension*. First and foremost a dimension is an aspect of some or most of the facts, and as such a set of values.

Additionally, dimensions can be structured. If the structure neither is embodied in the graph data itself nor can be derived from it but has to be provided from external sources instead (e.g., by joining it to the GeoNames dataset to obtain the relationship between cities, countries and continents), it is said to be *extrinsic*, otherwise the structure is *intrinsic* in the graph data. Regarding intrinsic dimensional structure we can further differentiate between *explicit*, if it is present in the form of connected vertices (e.g., a product category hierarchy), and *implicit*, if it can be derived from attribute values (e.g., extracting the day, month, and year components from an order date). Figure 3 shows examples for both explicit and implicit dimensional structure.

Dimensions are usually structured into levels; often more than one structure can be distinguished in a single dimension (e.g., a temporal dimension can be structured into days, months, and years as well as into days, months, and fiscal years). Each level can be identified by a unique name and described by a function mapping an aspect of facts to arbitrary values (e.g. vertices of type `order` to the month component of their delivery date attribute).

Definition 1 (Level). *A level* $l := (\text{Name}, \varphi)$ *is a tuple, where* $\varphi : \mathcal{G} \to X$ *is a unary function mapping a match of the corresponding dimension's seed pattern to an arbitrary value.*

Since the topmost level of a dimension has to produce a single element, we add an artificial root to the hierarchy. This is beneficial if all elements of a dimension should be placed in a single group.

Corollary 1 (Artificial Root). *The artificial root level* $\top = (\text{Top}, \varphi_\top)$ *yields a single root value* $\forall G \in \mathcal{G}.\, \varphi_\top(G) = \top$ *and is the topmost level, i.e.* $\forall l \in L.\, l \leq \top$.

Table 1. Two dimensions applicable to the previously selected facts.

Name	Seed Pattern	Levels
Nationality	$c	$c@nationality
Category	$p	1. Subcategory: $p-[@type='in']-> 2. Category: $p-[@type='in']->-[@type='part-of']-> 3. Super category: $p-[@type='in']->-[@type='part-of']->(2)

In addition to the totally ordered set of levels, a dimension specification (also uniquely identified by a name) consists of a seed pattern, which is applied to the facts. Table 1 illustrates these components for two dimensions that are applicable to the pairs of customer and review that were previously selected as facts. The syntax *alias*`@`*attribute* encodes the access to an attribute of the aliased vertex or edge, while `-[` *predicate* `]->(` *length* `)` denotes paths of a given length satisfying a given predicate.

Definition 2 (Dimension Specification). *A dimension specification* $d := (\text{Name}, S, L \cup \{\top\}, \leq)$ *is a tuple, where* $S : \mathcal{G} \to \mathcal{P}(\mathcal{G})$ *is the seed pattern,* $L \neq \emptyset$ *is a non-empty set of levels, and* $\leq \subseteq L \times L$ *is a total ordering of the levels. Level names are unique, i.e.* $\forall l \in L.\, (\exists m \in L : \text{Name}_m = \text{Name}_l) \iff m = l$.

Without loss of generality, the total ordering of the levels can be chosen such that the number of items per level decreases monotonically as the level increases. For example, while a (non-leap) year has 365 days, it has only 52 weeks and just 12 months.

Corollary 2 (Monotony). *Given two levels* $l, m \in L, l \neq m$*, it holds that* $l \leq m \implies |\varphi_m(\mathcal{G})| \leq |\varphi_l(\mathcal{G})|$.

The levels in a structured dimension often form hierarchies, meaning that seed pattern matches mapped to the same value in a lower level will also be mapped to the same value at a higher level. For example, all purchase orders recorded in January 2014 will be mapped both to `January` and to `2014`.

Corollary 3 (Hierarchy). *Given two levels* $l, m \in L, l \neq m$*, and two matches of the seed pattern* $G, F \in \mathcal{G}, G \neq F$*, it holds that*
$l \leq m \iff (\varphi_l(G) = \varphi_l(F) \implies \varphi_m(G) = \varphi_m(F))$.

3.3 Measures

A *measure* is derived from facts using arithmetic operations.

Definition 3 (Measure). *A measure* $m := (\text{Name}, f, \sigma)$ *is a tuple, where* $f : \mathcal{G} \to \mathbb{R}$ *is a function computing a numerical value for a fact and* $\sigma \in \{\text{SUM}, \text{AVG}, \text{MIN}, \text{MAX}, \ldots\}$ *is an aggregation function for combining numerical values when grouping facts.*

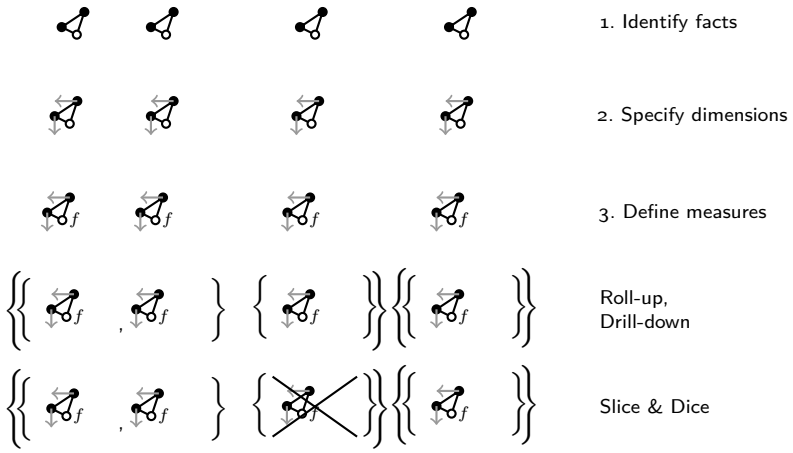

Fig. 4. Workflow for multidimensional graph analysis.

Measures are specified as part of the fact definition as an attribute of the newly introduced representative vertex. For example, in Figure 2 a measure for the average number of stars of a customer review is given as (Avg. #stars, $\texttt{\$r@stars}$, AVG), where the syntax for the specification of f means: starting from the vertex with the alias "$r", obtain the value of the attribute "stars".

4 Workflow

Building on the concepts defined in the previous section, we now introduce a workflow for performing multidimensional analytics on graph-structured data (cf. Figure 4). First, a *graph cube* has to be defined using the following steps:

1. **Identify facts.** By means of a graph pattern the user has to select the subgraphs of interest. For each fact a representative vertex that serves as a handle is created.
2. **Specify dimensions.** A dimension consists of at least one level. A level is a function that maps a subgraph to an arbitrary value. The connection between facts and mapping functions is achieved with the help of a seed pattern.
3. **Define measures.** By annotating the summarization rule used in the first step with a computation function and an aggregation function, a measure can be defined.

Definition 4 (Graph Cube). *A graph cube $c := (F, D, M)$ is an analytical scenario, where $F : \mathcal{G} \to \mathcal{P}(\mathcal{G})$ is the fact summarization rule, D is a set of dimension specifications, and M is a set of measures. Its granularity $\gamma : D \to L$ is initially set to the lowest level of each dimension: $\forall d \in D.\, \gamma(d) = \min(L_d)$.*

The dimensional structure can then be exploited to compute different groupings of facts and thereby transform the graph cube (or more specifically the measures associated with it).

Definition 5 (Roll-up/Drill-down). *The roll-up operation* $\uparrow\colon \Gamma \times D \to \Gamma$ *decreases the granularity for the specified dimension* $d \in D$ *by grouping facts according to the next higher level (where* $L_d = \{l_0, l_1, \ldots, l_n\}$ *and* $\gamma(d) = l_i$*). The drill-down operation* $\downarrow\colon \Gamma \times D \to \Gamma$ *increases the granularity by switching to the next lower level of the dimension. Derived granularities are defined as follows:*

$$\gamma_d^{\uparrow}(x) \coloneqq \begin{cases} l_{i+1} & \text{if } x = d \\ \gamma(x) & \text{otherwise} \end{cases} \qquad \gamma_d^{\downarrow}(x) \coloneqq \begin{cases} l_{i-1} & \text{if } x = d \\ \gamma(x) & \text{otherwise} \end{cases}$$

Definition 6 (Slice & Dice). *Given a set of level-predicate pairs, the function* $\text{filter} : C \times \mathcal{P}(L \times P) \to C$ *filters the fact base of a graph cube:* $\text{filter}((F, D, M), p) \coloneqq (F', D, M)$*, where* $F' = \{f | f \in F \wedge \forall (l, \lambda) \in p.\, \varphi_l(f) \models \lambda\}$.

If $|p| = 1$ a single predicate is applied to only one dimension and the operation is called *slice*, otherwise it is called *dice*. For example, to slice product reviews by German customers from the cube c, use $\text{filter}(c, \{(\text{Nationality}, \lambda = \text{“DE”})\})$. Now we have the basic concepts for multidimensional graph analytics in place.

5 Conclusions and Future Work

In this paper we presented a formal framework for graph analytics by mapping the well-known concepts *fact*, *dimension*, and *measure* from the multidimensional model to the property graph model. By relying on additive graph transformation rules as the means for selecting facts, we overcome the limitations of existing approaches: first, we can offer the user a graphical specification paradigm and avoid the introduction of a domain-specific language. Second, we can leverage the extensive research on efficient graph transformations from the past decades.

In the context of our research project SynopSys we will investigate the support of cross-cube measures for gaining insights into more complex topological aspects and anticipate additional OLAP operations based on our data model.

References

1. Rodriguez, M.A., Neubauer, P.: Constructions from Dots and Lines. Bulletin of the American Society for Information Science and Technology **36**(6) (2010) 35–41
2. Rudolf, M., Paradies, M., Bornhövd, C., Lehner, W.: SynopSys: Large Graph Analytics in the SAP HANA Database Through Summarization. In: Proc. GRADES. GRADES '13, ACM (2013) 16:1–16:6
3. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd edn. Wiley (June 2013)
4. Chen, C., Yan, X., Zhu, F., Han, J., Yu, P.S.: Graph OLAP: Towards Online Analytical Processing on Graphs. In: Proc. 8th ICDM, IEEE (2008) 103–112
5. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient Aggregation for Graph Summarization. In: Proc. SIGMOD. SIGMOD '08, ACM (2008) 567–580
6. Zhang, N., Tian, Y., Patel, J.M.: Discovery-Driven Graph Summarization. In: Proc. 26th ICDE, IEEE (2010) 880–891
7. Zhao, P., Li, X., Xin, D., Han, J.: Graph Cube: On Warehousing and OLAP Multidimensional Networks. In: Proc. SIGMOD, ACM (2011) 853–864