Graph-Driven Federated Data Management (Extended Abstract)

Sergi Nadal, Alberto Abelló, Oscar Romero
Universitat Politècnica de Catalunya
Barcelona, Spain
snadal|aabello|oromero@essi.upc.edu

Stijn Vansummerem
UHasselt - Hasselt University
Data Science Institute
Diepenbeek, Belgium
stijn.vansummeren@uhasselt.be

Panos Vassiliadis
University of Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

***Abstract*—Modern data analysis applications require the ability to provide on-demand integration of data sources while offering a user-friendly query interface. Traditional methods for answering queries using views, focused on a rather static setting, fail to address such requirements. To overcome these issues, we propose a full fledged, GLAV-based data integration approach based on graph-based constructs. The extensibility of graphs allows us to extend the traditional framework for data integration with view definitions. Furthermore, we also propose a query language based on subgraphs. We tackle query answering via a query rewriting algorithm based on well-known algorithms for answering queries using views. We experimentally show that our method yields good performance with no significant overhead.**

***Index Terms*—Data integration, query rewriting, GLAV mappings.**

I. Background and Motivation

Data wrangling is an iterative data exploration process to enable analysis. In contrast to *data warehousing* approaches, where data are materialized in a target schema tailored to a specific kind of analysis, *virtual data integration* systems play a key role on the exploration of a wealth of data that is yet to be integrated. As the popularity of wrangling systems grows, non-technical users face high-entry barriers on interacting with them, requiring queries to be written in technical languages such as Datalog or SPARQL. Additionally, the vast number of available heterogeneous datasets on the web pose several data integration challenges for contemporary wrangling demands [1]. In order to contribute towards resolving those challenges, we present a novel and full fledged approach to virtual integration using graphs as canonical data model for the whole process. Precisely, we present a framework for query answering over graphs mediating a set of heterogeneous data sources connected via *global-local-as-view* (GLAV) mappings.

The proposed framework (i.e., the *integration graph*) takes as building blocks Seth and Larson's reference architecture for federated databases [2], and adapts it to Lenzerini's data integration framework [3]. An integration graph (see Figure 1) contains all the metadata constructs composing a federated system. The main novelty of our approach is a query language based on coverings, or *contours*, of a graph representing the global schema. The proposed language does not require users to define join conditions, a task delegated to a rewriting algorithm. As opposed to classic methods where different data structures are maintained for schemata and mappings, our framework is grounded on graphs as unique data structure for all constructs. Besides the flexibility and ease of use that that brings to the wrangling process, using graphs to represent integration systems also brings performance benefits. Encoding all the required metadata (i.e., global schema, source descriptions, mappings and queries) in a single data structure simplifies the interoperability among them. This allows rewriting algorithms to query such metadata structures (e.g., mappings), bringing the ability to efficiently identify the relevant sources containing a query posed over the global schema.

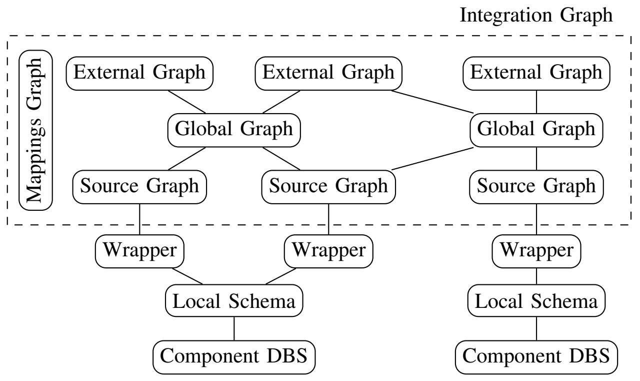

Fig. 1: Components in an Integration Graph, adapted from [2]

II. Problem Formulation

An integration graph $\mathcal{I}$ is a 4-tuple of edge-labeled directed graphs $\langle \mathcal{G}, \mathcal{S}, \mathcal{M}, \mathcal{E} \rangle$. The global graph, whose nodes are partitioned into two disjoint sets of concepts (C) and features (F), encode the conceptualization of the user's domain, while the source graph encodes, in a graph manner, wrappers (W) and their attributes (A). Then, a global query φ is represented as a connected subgraph of $\mathcal{G}$. This allows the definition of graph-based LAV mappings between $\mathcal{S}$ and $\mathcal{G}$. Precisely, for a wrapper w, a LAV schema mapping is a pair $\mathcal{M}(w) = \langle \varphi, \mathcal{F} \rangle$, where φ is a global query; and $\mathcal{F}$ is an injective function from A to F. Then, the external graph $\mathcal{E}$ encodes views (i.e., GAV mappings) together with the semantics of the expressions to compute derived features (i.e., operational expression trees).

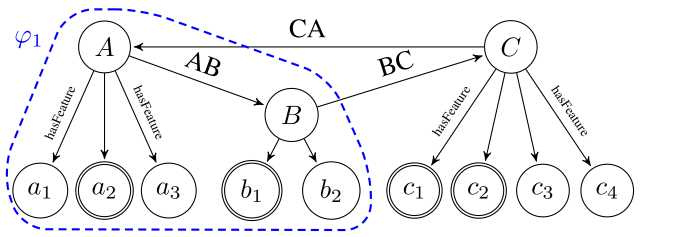

Fig. 2: Global graph and a query. Doubly circled features denote IDs. For clarity, some edge labels have been omitted.

Figure 2, depicts a global graph with three concepts. The global query φ_1 asks for all features from A and B.

We, then, define the notion of covering and minimal source queries. The former states that a source query ψ covers a global query φ if the union of the participating LAV mappings in ψ subsumes φ. Then, ψ is minimal w.r.t. φ if removing any wrapper from the source query generates a non-covering query. Using this notion, we consider rewriting algorithms over integration graphs (i.e., $\mathcal{R}_\mathcal{I}$) as functions with input a global query and output a set of source queries Ψ (i.e., unions of conjunctive queries). We say $\mathcal{R}_\mathcal{I}$ is *minimally-sound* if all source queries in $\mathcal{R}_\mathcal{I}(\varphi)$ are minimal. Likewise, we say $\mathcal{R}_\mathcal{I}$ is *minimally-complete* if for any source query ψ such that it is a rewriting of φ and is minimal, it holds that $\psi \in \mathcal{R}_\mathcal{I}(\varphi)$.

Problem statement. Rewriting queries φ over $\mathcal{I}$ reduces to finding a minimally-sound and minimally-complete rewriting algorithm $\mathcal{R}_\mathcal{I}$. We refer the reader to our full paper [4] for the detailed description and proofs of REWRITECQ, our proposal of such a minimally-sound and minimally-complete rewriting algorithm. REWRITECQ is inspired by the *bucket algorithm* for LAV mediation, which finds rewritings for each subgoal in the query, and stores them in buckets. Then, it finds a set of conjunctive queries such that each of them contains one conjunct from every bucket. In our case, concepts are analogous to buckets, however equi-join conditions must be automatically discovered. Hence, we first separately find rewritings that cover the requested concepts in φ to later find all possible minimal combinations among them.

III. Evaluation

In this section, we measure the performance of REWRITECQ and compare it to alternative approaches for answering queries using views. To assess our algorithms and facilitate their comparison to alternatives, we generate artificial data via a principled method. Precisely, we systematically generate synthetic experimental scenarios with different characteristics. Each scenario consists of a global graph, a set of wrappers, mappings, and a global query. For each combination of experimental variables, we generate an scenario and measure the processing time of REWRITECQ in seconds.

Comparison to alternatives on query rewriting. We compare our approach with the following state-of-the-art solutions for answering queries using views, whose source code is openly available: MiniCon [5] and Graal [6]. We compared the runtime for small values of $|F|$, as our tests showed that the alternatives struggled to manage a large number of features. Then, Figure 3 depicts the runtime comparison. First, we can observe that both alternatives have a much steeper exponential trend than ours. While we efficiently deal with 64 wrappers, MiniCon only manages to successfully execute around half of them. Graal fails to manage more than 10 wrappers. We believe the major performance drawback of such methods is the number of intermediate results they manage (i.e., candidate queries). Indeed, we have observed an exponential number of existential rules in their executions. Under these circumstances, exploration of the search space in a breadth-first search manner, as Graal does, becomes extremely costly. Oppositely, and considering we generate more solutions due to our rewriting semantics, thanks to the ability of querying the mappings, which are stored as graphs, we can select only relevant views in an incremental and more efficient manner.

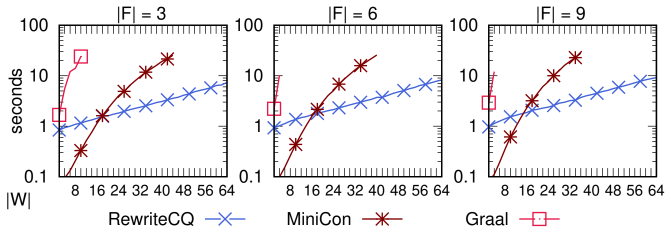

Fig. 3: Runtime evolution w.r.t. the number of wrappers ($|W|$) and alternative approach under comparison. Missing points denote that the execution of the alternatives timed out.

IV. Conclusions

In this paper, we provide an overview of a virtual data integration system grounded on graphs. We advocate that such unique, and widely accepted, formalism allows non-technical users to perform exploratory tasks, such as data wrangling. On top of that, the flexibility of graphs enables the extensibility of the current rewriting algorithm. For example, to jointly consider aggregations when running the rewriting algorithm.

Acknowledgements. Sergi Nadal is partly supported by the Spanish Ministerio de Ciencia e Innovación, as well as the European Union - NextGenerationEU, under project FJC2020-045809-I / AEI/10.13039/501100011033.

References

[1] T. Furche, G. Gottlob, L. Libkin, G. Orsi, and N. W. Paton, “Data wrangling for big data: Challenges and opportunities,” in *EDBT*, 2016.

[2] A. P. Sheth and J. A. Larson, “Federated database systems for managing distributed, heterogeneous, and autonomous databases,” *ACM Comput. Surv.*, vol. 22, no. 3, pp. 183–236, 1990.

[3] M. Lenzerini, “Data integration: A theoretical perspective,” in *PODS*, 2002.

[4] S. Nadal, A. Abello, O. Romero, S. Vansummeren, and P. Vassiliadis, “Graph-driven federated data management,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[5] R. Pottinger and A. Y. Halevy, “Minicon: A scalable algorithm for answering queries using views,” *VLDB Journal*, vol. 10, no. 2-3, pp. 182–198, 2001.

[6] J. Baget, M. Leclère, M. Mugnier, S. Rocher, and C. Sipieter, “Graal: A toolkit for query answering with existential rules,” in *RuleML*, 2015.