On Directly Mapping Relational Databases to Property Graphs

Radu Stoica[1], George Fletcher[1], and Juan F. Sequeda[2]

[1] Eindhoven University of Technology, the Netherlands
{r.a.stoica@student., g.h.l.fletcher@}tue.nl
[2] Capsenta, Austin, Texas, USA
juan@capsenta.com

Abstract. Much of the data found in practice resides in relational DBs. However, many contemporary analytical tasks are performed on graphs. Property graphs are currently one of the most prevalent data models for graph data management in industry. Therefore, a key challenge is to understand the fundamental relationships between relational databases and property graph databases. This paper reports our ongoing work towards understanding these relationships by proposing R2PG-DM, a direct mapping of relational databases to property graphs. Given a relational database schema and instance, a direct mapping generates a corresponding property graph instance. The semantics of our mapping is defined using Datalog. Our work is inspired by existing approaches for direct mappings of relational databases into earlier graph data models. Future work is to study our mapping with respect to fundamental properties such as information and query preservation.

1 Introduction

A major class of contemporary data analytics focuses on gaining insights from the rich complex patterns found in graph-structured data collections such as social, communication, financial, biological, and mobility networks [1]. For example, investigative journalists have recently found, through graph analytics, surprising social relationships between executives of companies within the Offshore Leaks financial social network data set, linking company officers and their companies registered in the Bahamas.[3] Property graphs (PG) are currently one of the most prevalent data models for the management of such data in industry [1]. A PG is an edge- and node-labeled directed multigraph where both edges and nodes have associated sets of properties, i.e., key-value pairs.

The Offshore Leaks PG was constructed as a mapping from relational database (RDB) sources.[4] Indeed, much of the data found in practice resides in RDBs. Therefore, a key challenge is to understand the fundamental relationships between RDBs and PGs. A crucial first step in exploratory graph analytics on

[3] *International Consortium of Investigative Journalists.* https://offshoreleaks.icij.org/
[4] https://www.icij.org/blog/2013/06/how-we-built-offshore-leaks-database/

RDBs is to transform the database into a PG. Only then can the basic graph structure be explored and new relationships discovered using contemporary graph database systems.

This motivates the study of direct mappings from RDBs to PGs. A *direct mapping* is a transformation from RDBs to PGs which (1) is domain and schema-independent, i.e., works regardless of the source database schema and instance, and (2) transforms the content of the source instance into a target instance, i.e., given a RDB instance generates a corresponding PG instance.

State of the art. Most approaches to graph analytics over relational data extract graphs as user-specified views on relational data, e.g., [3,8]. This requires, however, that the user already fully understands the desired graph view, which is overly restrictive in the common case of exploratory graph analytics.

The study of direct mappings from relational to property graphs is not as well-developed. Of the small handful of approaches, the focus has been on optimized layout for efficient query processing or mappings which are lossy or obfuscate the input RDB schema [4,5,7,9]. Furthermore, all current direct mappings are defined procedurally (i.e., defined with pseudo-code), making it difficult to formally reason about their correctness and other basic properties.

Our contributions. In this short note, we report on our work-in-progress on R2PG-DM, a declarative direct mapping from RDB to PG. R2PG-DM losslessly transforms both the source instance and schema while also intuitively preserving the original structure of the input RDB. Our work is inspired by the approach of Sequeda et al. to direct mappings from relational to RDF graphs, the W3C standard for sharing graph-structured data on the web [6]. We present R2PG-DM by example and outline the basic research questions we are currently investigating in our study of the connections between RDB and PG.

2 RDBs, PGs, and direct mappings

Let $\mathcal{D}$ be an enumerable set of *data values* containing the special value null, and let $\mathcal{A}$ be an enumerable set of *attribute names* containing the special attribute tid. An RDB *schema* is a triple $\mathbf{S} = (R, att, \Sigma)$, where R is a finite set of *relation names*, att is a function assigning to each $r \in R$ a finite set $att(r) \subseteq \mathcal{A} \setminus \{\mathsf{tid}\}$, and Σ is a finite set of *primary* and *foreign keys* over R and att.[5] For $r \in R$, an *r-tuple* is a function $t : att(r) \cup \{\mathsf{tid}\} \to \mathcal{D}$ such that $t(\mathsf{tid}) \neq \mathsf{null}$. An *instance* I of $\mathbf{S}$ is an assignment to each $r \in R$ of a finite set $\mathsf{I}(r)$ of r-tuples satisfying Σ, such that for distinct $t, t' \in \bigcup_{r \in R} \mathsf{I}(r)$ it holds that $t(\mathsf{tid}) \neq t'(\mathsf{tid})$. An example schema and instance is given in Figure 1 (top left).

A *property graph* is a structure (V, E, e, ℓ, p) where V is a finite set of *vertices*, E is a finite set of *edges*, e is a function assigning an ordered pair of vertices to each edge (i.e., the source and target vertices of the edge, resp.), ℓ assigns a finite set of *labels* to each vertex and edge (from some domain of labels), and p assigns a finite set of *key-value pairs* to each vertex and edge. An example property graph is given in Figure 1 (bottom left).

[5] https://en.wikipedia.org/wiki/Foreign_key

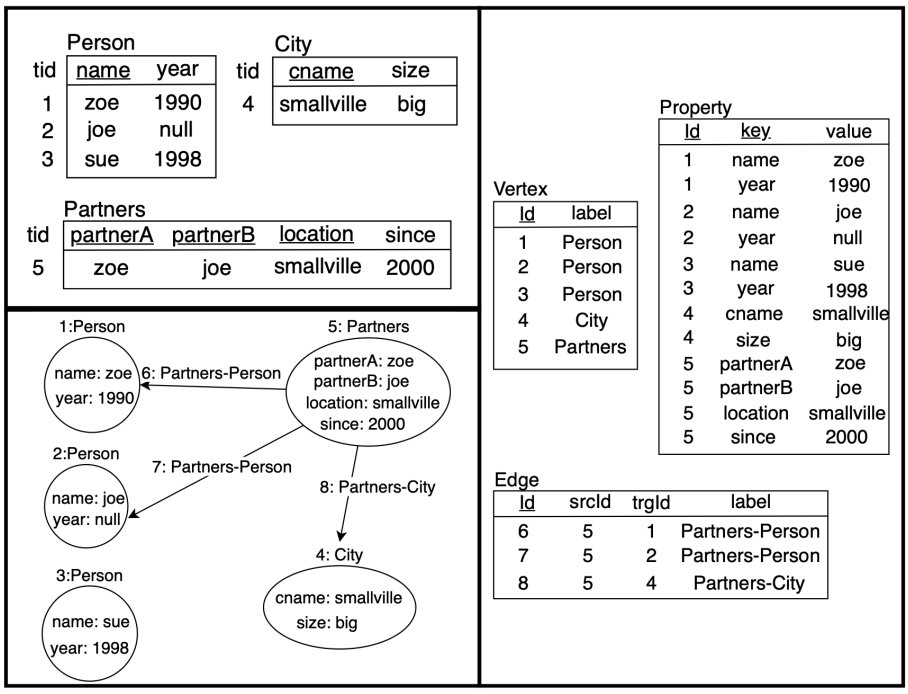

Fig. 1. (top left) An RDB $(\mathbf{S}, \mathsf{I})$, where primary key attributes are underlined, partnerA and partnerB of Partners are foreign keys to name of Person, and location of Partners is a foreign key to cname of City. (bottom left) The property graph R2PG-DM($\mathbf{S}, \mathsf{I}$) and (right) its representation with predicates *Vertex*, *Edge*, and *Property*.

Let $\mathcal{PG}$ denote the set of all PGs and $\mathcal{RDB}$ denote the set of all pairs $(\mathbf{S}, \mathsf{I})$ where $\mathbf{S}$ is an RDB schema and I is an instance of $\mathbf{S}$. A *direct mapping* is a total function from $\mathcal{RDB}$ to $\mathcal{PG}$.

3 The R2PG-DM direct mapping

We next informally present R2PG-DM, a direct mapping which addresses the shortcomings of current solutions discussed in Section 1. With R2PG-DM, RDB relations are interpreted as classes of "things" (i.e., labeled vertices) and foreign-key relationships are interpreted as edges between these things. In particular:

- each input r-tuple t is represented in the output graph by a vertex v (identified by $t(\mathsf{tid})$), which is labeled with r, the name of the relation to which t belongs; furthermore, v has key-value pair $(a, t(a))$, for each $a \in att(r)$.
- for each foreign key relationship $f \in \Sigma$ (from, say, relation r to relation s), for each pair of tuples $t \in \mathsf{I}(r)$ and $t' \in \mathsf{I}(s)$ participating in f, this relationship is represented by an edge with label r-s from vertex v_t to vertex $v_{t'}$, representing t and t', respectively.

Similarly, as there currently does not exist a standard PG schema language, the input schema is also fully represented in the output PG, e.g., each relation and each attribute of a relation is represented by a vertex, and foreign keys are represented by edges between the relevant attributes of relations, etc. Figure 1 illustrates an RDB database $(\mathbf{S}, \mathsf{I})$ and PG translation R2PG-DM$(\mathbf{S}, \mathsf{I})$. We omit here the translation of $\mathbf{S}$ to PG, and only illustrate the translation of I.

Clearly, each component of the R2PG-DM mapping can be specified by a declarative non-recursive Datalog query [2] over the source DB represented using a fixed set of predicates (see Section 4.1 "Storing relational databases" of Sequeda et al. [6]), with the target PG represented by three predicates $Vertex$, $Edge$, and $Property$ capturing the vertices, edges, and the key-value properties associated with vertices and edges, respectively. This is illustrated in Figure 1 (right).

4 Ongoing research

Our ongoing work proceeds along three lines. First, we are proceeding with a formal study of R2PG-DM, establishing basic properties such as information and query preservation [6]; we hypothesize that R2PG-DM generates a graph that is isomorphic as a graph generated by the direct mapping of [6]. Moreover, we are studying how to extend R2PG-DM in order to generate a graph with properties on edges, taking full advantage of the data model. Second, we are developing practical tools for efficient and scalable R2PG-DM, to be made available as open source code. Third, we aim to extend the R2PG-DM approach to support customized mappings for the cases where there is a schema defined on the target instance (e.g., mapping the relational schema $\mathbf{S}$ to an equivalent PG schema). This builds upon ongoing efforts towards standards for PG schema languages.[6]

References

1. Angela Bonifati, George Fletcher, Hannes Voigt, Nikolay Yakovets. *Querying Graphs.* Morgan & Claypool, 2018.
2. Todd J. Green et al. Datalog and recursive query processing. *Foundations and Trends in Databases* 5(2):105-195, 2013.
3. Mohamed S. Hassan et al. Extending in-memory relational database engines with native graph support. In *EDBT 2018*, pages 25-36.
4. Ognjen Orel, Slaven Zakošek, Mirta Baranović. Property oriented relational-to-graph database conversion. *Automatika* 57(3): 836-845, 2016.
5. Subhesh Pradhan, Sharma Chakravarthy, Aditya Telang. Modeling relational data as graphs for mining. In *COMAD 2009*.
6. Juan F. Sequeda, Marcelo Arenas, Daniel P. Miranker. On directly mapping relational databases to RDF and OWL. In *WWW 2012*, pages 649-658.
7. Roberto De Virgilio, Antonio Maccioni, Riccardo Torlone. R2G: a tool for migrating relations to graphs. In *EDBT 2014*, pages 640-643.
8. Konstantinos Xirogiannopoulos, Amol Deshpande. Extracting and analyzing hidden graphs from relational databases. In *SIGMOD 2017*, pages 897-912.
9. Kang Min Yoo, Sungchan Park, Sang-Goo Lee. RDB2Graph: a generic framework for modeling relational databases as graphs. In *JIST 2014*, pages 148-151.

[6] `https://www.w3.org/Data/events/data-ws-2019/`